Лекция 2. Детерминированные конечные автоматы

Главные вопросы, которые мы обсуждаем, представлены на СЛАЙДЕ 1. Рассмотрим более подробно КА и введем новое понятие – **регулярные языки**.

2.1 Введение

Как указывалось ранее, КА состоит из множества состояний и некоторой логики управления, которое переводит автомат из одного состояния в другое в зависимости от получаемых извне входных данных (или **сигналов**). По типу управления принято разделять автоматы на детерминированные, или ДКА (автомат может находиться только в одном состоянии в определенный момент времени) и недетерминированные, или НКА (автомат может находиться более, чем в одном состоянии). Добавление недетерминизма не позволяет определять языки, которые нельзя было бы определить с помощью ДКА. Однако НКА могут эффективно решать ряд прикладных задач на языках программирования высокого уровня. Далее мы покажем, как можно преобразовать НКА в ДКА, который, в свою очередь, может быть выполнен (моделирован) на вычислительной машине.

Также далее будут определены расширенные НКА (РНКА), у которых имеется возможность перехода из состояния в состояние **спонтанно**, по пустой строке на входе. РНКА описывают только регулярные языки и полезны при изучении регулярных выражений и доказательстве их эквивалентности автоматам.

2.2 Детерминированные конечные автоматы

Сразу оговоримся, что в этой части лекции термины КА (FA – *Finite Automaton*) и ДКА (DFA – *Deterministic Finite Automation*) будут использоваться как взаимозаменяемые.

Если A – это имя ДКА, тогда он состоит из следующих компонентов (СЛАЙД 2):

$$A = (Q, \Sigma, \delta, q_0, F),$$

где Q – конечное множество состояний, Σ – конечное множество входных символов, δ – функция переходов, q_0 – начальное состояние ($q_0 \in Q$), F – множество заключительных (допускающих) состояний ($F \subseteq Q$).

В предыдущем разделе лекционного курса при неформальном рассмотрении КА мы изображали его в виде ориентированного графа. В этом смысле функция переходов δ – это дуги графа, соединяющие состояния. Формально же, это функция двух аргументов $\delta(q,a)$, где q – состояние, a – входной символ (или сигнал). Значением является новое состояние p, для которого существует дуга, отмеченная a и ведущая из q в p.

Главное, что пока следует уяснить – это понять как ДКА решает вопрос допустимости последовательности входных символов. Язык ДКА – это набор всех его допустимых строк.

Предположим $a_1a_2...a_n$ – это последовательность входных символов, и ДКА начинает работу в состоянии q_0. Для того чтобы найти состояние, в которое A перейдет после обработки символа a_1, нужно обратиться к функции переходов δ. Допустим, $\delta(q_0,a_1) = q_1$. Аналогично находятся последующие состояния $q_2,...,q_n$, где $\delta(q_{i-1},a_i) = q_i$ для каждого i. Если при этом q_n принадлежит множеству F, то входная строка $a_1a_2...a_n$ **допускается**, в

противном случае – **отвергается** (не допускается). СЛАЙД 3.

Пример 4.

Определим формально ДКА, допускающий строки из 0 и 1, которые содержат в себе подстроку 01 (СЛАЙД 4). Этот язык можно описать следующим образом:

L_A = {w | w имеет вид $x01y$, где x и y – строки из 0 и 1} или альтернативно:

L_A = {$x01y$ | x и y – строки из 0 и 1}.

Примеры строк этого языка: 01, 11010, 1011101. Не принадлежат этому языку, например, следующие строки: 0, 1, 1100 и ε.

Алфавитом входных символов языка L_A является Σ = {0, 1}. У автомата A имеется некоторое множество состояний. Один из элементов этого множества, скажем, q_0 – начальное состояние. Чтобы решить, содержит ли входная строка подцепочку 01, автомат должен помнить следующие важные факты относительно уже прочитанных элементов строки.

1. Была ли прочитана последовательность 01? Если была – всякая читаемая далее последовательность допустима, и ДКА будет находиться в заключительных состояниях.

2. Если последовательность 01 еще не была прочитана, то был ли на предыдущем шаге считан символ 0? Если был, и на текущем шаге считывается символ 1, то последовательность 01 будет прочитана, и после этого КА будет находиться только в заключительных состояниях.

3. Если последовательность 01 еще не была прочитана, и на предыдущем шаге на вход ничего не подавалось (состояние начальное) либо считан символ 1? В этом случае КА не переходит в допускающие состояния до тех пор, пока не будут считаны символы 0 и сразу за ним 1.

Каждый из этих фактов можно представить как некоторое состояние. Так (3) соответствует состояние q_0. Если A действительно находится в начале процесса, нужно один за другим прочитать символы 0 и 1. Однако если в состоянии q_0 считывается 1, это нисколько не приближает A к ситуации, когда прочитана нужная подстрока. Значит, придется оставаться в q_0. Итак, $\delta(q_0,1) = q_0$.

Следуем дальше. Если в состоянии q_0 читается 0, то мы попадаем в (2). Это означает, что подстрока 01 еще не прочитана, а 0 – прочитан. Обозначим это состояние как q_1, а переход будет иметь вид $\delta(q_0,0) = q_1$.

В состоянии q_1 при чтении 0 мы попадаем в аналогичную ситуацию: 01 еще не прочитана, но уже прочитан 0, и ожидается 1. Эта ситуация описывается состоянием q_1 поэтому получим $\delta(q_1,0) = q_1$. Если в этом состоянии считывается 1, то очевидно, что во входной строке непосредственно за 0 идет 1. Таким образом, можно перейти в заключительное состояние q_2. Это соответствует (1), т.е. $\delta(q_1,1) = q_2$.

Осталось построить переходы в состоянии q_2. В этой ситуации требуемая подстрока уже прочитана, и автомат будет находиться в этом же состоянии, т.е. $\delta(q_2,0) = \delta(q_2,1) = q_2$.

Теперь ясно, что $Q = \{q_0,q_1,q_2\}$, $F = \{q_2\}$, а функция переходов описана выше.

2.3 Более простые представления ДКА

Для многих определение ДКА как пятерки элементов с детальным описанием функций переходов не является удобным. Чаще пользуются двумя другими способами описания автомата

1. Диаграмма переходов, представляющая собой граф, примеры которых мы приводили в предыдущем разделе.

2. Таблица переходов, дающая табличное представление функции δ, из которой очевидны состояния и входной алфавит.

Диаграмма переходов для ДКА есть граф, определяемый следующим образом (СЛАЙД 5):

а) любому состоянию из Q соответствует некоторая вершина;

б) пусть $\delta(q,a) = p$ для некоторого q из Q и входного символа q из Σ. Тогда

диаграмма должна содержать дугу из q в p, отмеченную a. Если существует несколько символов, переводящих автомат из q в p, то диаграмма переходов может содержать одну дугу, отмеченную списком этих символов;

в) диаграмма содержит стрелку в начальное состояние. Стрелка не выходит ни из какого состояния;

г) вершины, соответствующие заключительным состояниям отмечаются двойным кружком. Не являющиеся заключительными состояния изображаются одинарным кружком.

Пример 5.

На СЛАЙДЕ 6 изображена диаграмма переходов для ДКА, построенного в примере 4. Видны три вершины (по числу состояний). Из каждого состояния выходят две дуги: одна отмечена 0, вторая – 1, но для состояния q_2 дуги объединены. Каждая из дуг соответствует одному из фактов для δ.

Таблица переходов представляет собой обычное табличное представление функции вроде δ, которая двум аргументам ставит в соответствие одно значение. Строки таблицы соответствуют состояниям, столбцы – входным символам. На пересечении строк для состояния q и столбца для символа a находится состояние $p = \delta(q,a)$. СЛАЙД 7.

Пример 6.

На СЛАЙДЕ 8 представлена таблица переходов для ДКА, построенного в примере 4. Очевидны и другие особенности таблицы переходов. Начальное состояние отмечено стрелкой, а допускающее – звездочкой. Поскольку заглавные литеры строк и столбцов задают множества состояний и символов, то у нас есть вся информация, необходимая для однозначного описания нашего ДКА.

Выше было нестрого обосновано утверждение о том, что любой ДКА определяет некоторый язык как множество всех строк, приводящих КА из начального состояния в одно из заключительных. В терминологии диаграмм переходов это множество меток вдоль всех путей, ведущих из начального состояния в любое заключительное. Для примера 5 это метки из q_0 в q_2.

Чтобы дать строгое определение языка ДКА, мы должны сначала определить **расширенную функцию переходов** (далее – РФП). Она описывает ситуацию, когда, начиная с произвольного состояния, отслеживается произвольная последовательность входных символов. Для нашей функции переходов δ РФП мы обозначим $\hat{\delta}$. Она состоянию q и строке w ставит в соответствие новое состояние p, в которое КА попадает из состояния, обработав входную последовательность w. Мы можем определить $\hat{\delta}$ индукцией по длине входной строки следующим образом (СЛАЙД 9).

Итак, $\hat{\delta}(q,\varepsilon) = q$, т.е. находясь в некотором состоянии и не читая вход, КА остается в том же состоянии.

Пусть w – строка вида xa, где a – последний символ в строке, а x – строка, состоящая из символов w, но без последнего символа (это ведь наш замечательный a). Скажем, w = 0100 разбивается на x = 010 и a = 1. Тогда получим:

$$\hat{\delta}(q,w) = \delta\left(\hat{\delta}(q,x),a\right)$$

Поясним себе это выражение. Для того чтобы найти $\hat{\delta}(q,w)$ мы сперва находим $\hat{\delta}(q,x)$. Это такое состояние, в которое КА переходит после обработки всех символов строки w, кроме последнего. Предположим, что это состояние p, т.е. $\hat{\delta}(q,x) = p$. Тогда $\hat{\delta}(q,w)$ – это состояние, в которое КА переходит из p при чтении символа a – это последний символ w. Таким образом, $\hat{\delta}(q,w) = \delta(p,a)$, что нам и требовалось.

Пример 7.

Теперь мы попробуем получить ДКА, который допускает язык *L*. СЛАЙД 10.

L = {*w* | *w* содержит четное число 0 и четное число 1}.

Как и в предыдущем примере для решения задачи будут использоваться состояния КА. В нашем случае – для получения количества 0 и 1. Подсчет ведется по модулю 2, иначе говоря, состояния в каждый момент времени запоминают четное или нечетное число 0 и 1. Следовательно, существуют четыре состояния, описываемые следующим образом.

q_0 – прочитано четное число 0 и четное число 1.

q_1 – прочитано четное число 0 и нечетное число 1.

q_2 – прочитано нечетное число 0 и четное число 1.

q_3 – прочитано нечетное число 0 и нечетное число 1.

Интересно, что состояние q_0 является и начальным, и единственным заключительным. Начальным оно является потому, что до чтения всех элементов входной строки, количество прочитанных 0 и 1 равно нулю, а нуль – четное число. Это состояние заключительное, т.к. в точности описывает условие принадлежности языку *L* последовательностей из 0 и 1.

Опишем теперь автомат *A*:

$A = (\{q_0, q_1, q_2, q_3\}, \{0,1\}, \delta, q_0, \{q_0\})$.

Функция переходов в форме диаграммы показана на СЛАЙДЕ 11.

Данный ДКА можно представить таблицей переходов, показанной на СЛАЙДЕ 12.

Однако нам ведь нужно не только построить ДКА, но и показать с его помощью, как строится функция $\hat{\delta}$ по функции переходов δ. Допустим, на вход подается строка 110101. Таким образом, мы ожидаем, что $\hat{\delta}(q_0, 110101) = q_0$. Постараемся доказать это.

Для проверки требуется найти $\hat{\delta}(q_0, w)$ для всех префиксов *w* строки 110101, начиная с ε. Результат будет выглядеть следующим образом (СЛАЙД 13).

- $\hat{\delta}(q_0, \varepsilon) = q_0$.
- $\hat{\delta}(q_0, 1) = \delta(\hat{\delta}(q_0, \varepsilon), 1) = \delta(q_0, 1) = q_1$.
- $\hat{\delta}(q_0, 11) = \delta(\hat{\delta}(q_0, 1), 1) = \delta(q_1, 1) = q_0$.
- $\hat{\delta}(q_0, 110) = \delta(\hat{\delta}(q_0, 11), 0) = \delta(q_1, 0) = q_2$.
- $\hat{\delta}(q_0, 1101) = \delta(\hat{\delta}(q_0, 110), 1) = \delta(q_2, 1) = q_3$.
- $\hat{\delta}(q_0, 11010) = \delta(\hat{\delta}(q_0, 1101), 0) = \delta(q_3, 0) = q_1$.
- $\hat{\delta}(q_0, 110101) = \delta(\hat{\delta}(q_0, 11010), 1) = \delta(q_1, 1) = q_0$.

2.4 Язык ДКА

Теперь мы можем определить язык, допускаемый ДКА. Для автомата *A* этот язык обозначим как *L*(*A*) и определим его следующим образом (СЛАЙД 14).

L(*A*) = { *w* | $\hat{\delta}(q_0, w)$ принадлежит *F* }.

Таким образом, язык – это множество строк, приводящих ДКА из состояния q_0 в одно из заключительных состояний. Если язык *L* есть *L*(*A*) для некоторого автомата *A*, то говорят, что *L* является **регулярным языком**.

Пример 8.

Выше упоминалось, что если *A* – ДКА из примера 4, то *L*(*A*) – это множество строк из 0 и 1, которые содержат подстроку 01. Одна из возможных программных реализаций

показана на СЛАЙДАХ 15-17.

Если же A – ДКА из примера 7, то $L(A)$ – это множество строк из 0 и 1, которые содержат четное число 0 и четное число 1. Одна из возможных программных реализаций показана на СЛАЙДАХ 18-20.

2.5 Проверка эквивалентности состояний ДКА

Попробуем разобраться, как два различных состояния ДКА, обозначим их p и q, можно заменить одним, работающим одновременно как они. Состояния p и q **эквивалентны**, если для всех входных строк w состояние $\hat{\delta}(p,w)$ является заключительным тогда и только тогда, когда состояние $\hat{\delta}(q,w)$ является заключительным. СЛАЙД 21.

Проще говоря, эквивалентные состояния неразличимы, если проверить, допускает ли КА заданную входную строку, начиная работу в одном из них. Эти состояния не обязаны совпадать. Главное – оба они являются заключительными либо оба они не являются заключительными.

Если эти состояния не являются эквивалентными, то они **различимы**, т.е. существует, по меньшей мере, одна строка, для которой одно из состояний заключительное, а другое – нет.

Пример 9.

Рассмотрим ДКА, приведенный на СЛАЙДЕ 22. Функцию переходов обозначим символом δ. Некоторые пары состояний эквивалентными не являются, например, C и G, поскольку C – заключительное, а G – нет. Пустая строка различает эти состояния, потому что $\hat{\delta}(C,\varepsilon)$ – заключительное состояние, а $\hat{\delta}(G,\varepsilon)$ – нет.

Состояния A и G различить с помощью ε невозможно, т.к. они оба не заключительные. Символ 0 их не различает, т.к. по входу 0 КА переходит в B и G соответственно, а они оба не заключительные. С другой стороны, строка 01 позволит их различить, поскольку $\hat{\delta}(A,01)=C$ (это заключительное состояние), а $\hat{\delta}(G,01)=E$ (незаключительное). Для доказательства неэквивалентности достаточно найти хотя бы одну входную строку, переводящую КА из проверяемых состояний (у нас A и G) в такие состояния, одно из которых является заключительным, а другое – нет.

Смотрим на состояния A и E. Ни одно из них не является заключительным, так что пустая строка не различает их. По входу 1 в обоих состояниях КА переходит в состояние F. Следовательно, ни одна входная строка, начинающаяся с 1, не может различить их, т.к. $\forall x|\hat{\delta}(A,1x)=\hat{\delta}(E,1x)$. В качестве альтернативы рассмотрим поведение в тех же состояниях по входам, начинающимся с 0. Из состояния A автомат переходит в B, а из E – в H. Оба они не заключительные, так что строка 0 не отличит A и E. Взглянув на состояния B и H, мы увидим, что они не помогают: по входу 0 они переходят в G, по 1 – в C. Таким образом, ни одна входная строка, начинающаяся с 0, не может различить A и E. Значит, они являются эквивалентными, что от них и требовалось.

Из примера должно быть видно, что для поиска эквивалентных состояний, нужно выявить все пары различимых состояний. Кому-то может показаться странным, но если найдены все пары состояний, различимых в соответствии с описываемой далее методикой, то те пары состояний, которые найти не удалось, являются эквивалентными. Этот метод, называемый **алгоритмом заполнения таблицы**, производит рекурсивное обнаружение пар различимых состояний ДКА. СЛАЙД 23.

(1) Итак, если состояние p – заключительное, а q – не заключительное, то пара состояний $\{p, q\}$ различима.

(2) Пусть p и q – состояния, по входному символу a переходящие в различимые состояния $r=\delta(p,a)$ и $s=\delta(q,a)$. Тогда пара состояний $\{p, q\}$ различима по очевидной

причине.

Пример 10.

Покажем работу алгоритма заполнения, взяв ДКА из примера 9. **Таблица неэквивалентности** заполняется так, что в ячейках явно указываются пары различимых состояний (например, символом *x*). Пустые ячейки соответствуют эквивалентным состояниям. Вначале все состояния полагаются эквивалентными.

Поскольку *C* у нас единственное заключительное состояние, то воспользовавшись (1), в каждую пару, содержащую *C*, записывается *x*. Имея сведения о некоторых парах различимых состояний, можно найти другие. В частности, по причине различимости пары $\{C, H\}$, в состояния *E* и *F* по входному символу 0 переходят в *H* и *C*, то пару $\{E, F\}$ мы тоже можем считать различимой. Таблица неэквивалентности приведена на СЛАЙДЕ 24.

Для всех пар, за исключением $\{A, G\}$, различимость выясняется просто. Просматриваются все переходы из каждой пары состояний по символам 0 или 1, и в результате обнаруживается, что из одного состояния есть переход в *C*, из другого – нет. В случае пары $\{A, G\}$ различимость выявляется так. По символу 1 они переходят в *F* и *E*, а различимость этих состояний уже установлена.

А вот обнаружить другие пары различимых состояний оказывается невозможно. Значит, пары $\{A, E\}$, $\{B, H\}$, $\{D, F\}$ являются эквивалентными. Докажем, например, неразличимость пары $\{A, E\}$. По входному символу 0 они переходят в *B* и *H*, но про эту пару неизвестно, различима она или нет. По входному символу 1 они оба переходят в *F*, то различить их таким способом не удастся. Другие две пары различить также не получится из-за того, что у них одинаковые символы и по 0, и по 1. Значит, алгоритм заполнения останавливается на приведенной таблице, и корректно определяет эквивалентные и различимые состояния.

Теорема 2.1. Если два состояния не различаются с помощью алгоритма заполнения, то они эквиваленты.

Доказательство. Рассмотрим ДКА $A = (Q, \Sigma, \delta, q_0, F)$. Допустим, утверждение теоремы неверно, а стало быть, существует, по меньшей мере, одна пара состояний (p, q), для которой выполняются два условия (СЛАЙДЫ 25-28):

(1) Состояния *p* и *q* различимы.

(2) Алгоритм заполнения таблицы не может обнаружить различимость *p* и *q*. Эта пара называется **плохой парой**.

Допустим, у нас есть плохие пары. Среди них должны быть различимые с помощью кратчайших строк, различающих плохие пары. Пусть наша пара $\{p, q\}$ – плохая, $w = a_1 a_2 ... a_n$ – кратчайшая из всех строк, различающих *p* и *q*. Тогда только одно из состояний $\hat{\delta}(p, w)$ и $\hat{\delta}(q, w)$ является заключительным.

Если некоторая пара состояний различается с помощью ε, то ее можно обнаружить, выполнив первую часть алгоритма заполнения. Значит, $w \neq \varepsilon$, т.е. $n \geq 1$.

Далее мы рассматриваем состояния $r = \delta(p, a_1)$ и $s = \delta(q, a_1)$. Их можно различить строкой $a_2 ... a_n$, т.к. она переводит КА из состояний *r* и *s* в состояния $\hat{\delta}(p, w)$ и $\hat{\delta}(q, w)$. Замечаем, что строка, отличающая *r* от *s*, короче любой строки, различающей плохую пару. Значит, $\{r, s\}$ – не является плохой парой, и алгоритм заполнения так или иначе обнаружит, что эти состояния различимы.

Однако вторая часть алгоритма не остановится, пока не придет к выводу, что состояния *p* и *q* различимы. Мы пришли к противоречию с предположением о наличии плохих пар, т.е. их нет. Следовательно, любую пару различимых состояний можно обнаружить алгоритмом заполнения, и наша теорема доказана.

Примеры двух эквивалентных ДКА приведены на СЛАЙДЕ 29.

2.6 Минимизация ДКА

Важным следствием проверки эквивалентности состояний является возможность минимизации ДКА. Для каждого ДКА можно найти эквивалентный ему ДКА с наименьшим числом состояний, и для заданного языка существует единственный минимальный ДКА (далее – МДКА) с точностью до выбираемых обозначений.

Основная идея минимизации заключается в том, что эквивалентность состояний позволяет объединять их в блоки.

(1) Все состояния в блоке эквивалентные.

(2) Любые два состояния из разных блоков неэквивалентны.

Пример 11.

Рассмотрим таблицу из примера 10 для автомата из примера 9. Эти состояния разбиваются на блоки следующим образом: $\{A, E\}$, $\{B, H\}$, $\{C\}$, $\{D, F\}$, $\{G\}$. Каждая пара эквивалентных состояний помещается в отдельный блок, а состояния, отличные от других, образуют отдельные блоки. СЛАЙД 30.

Теорема 2.2. Эквивалентность состояний транзитивна. Т.е. если для некоторого ДКА состояние p эквивалентно q, а состояние q – r, то состояния p и r также эквивалентны.

Доказательство. Предположим, что $\{p, q\}$ и $\{q, r\}$ являются парами эквивалентных состояний, а пара $\{p, r\}$ – различима. Тогда должна существовать строка w, для которой одно из состояний $\hat{\delta}(p, w)$ и $\hat{\delta}(r, w)$ является заключительным. Используя симметрию, предполагаем, что $\hat{\delta}(p, w)$ – заключительное. Если при этом $\hat{\delta}(q, w)$ – заключительное состояние, то пара $\{q, r\}$ различима. Если же $\hat{\delta}(q, w)$ – не заключительное состояние, то по похожей причине пара $\{p, q\}$ различима. Полученное противоречие доказывает неразличимость пары $\{p, r\}$. Это значит, что состояния p и r эквиваленты. СЛАЙДЫ 31-32.

Используя эту теорему, можно обосновать простой алгоритм разбиения состояний. Для любого состояния q создается блок, состоящий из q и его эквивалентов. Нужно доказать, что полученные блоки образуют разбиение множества, т.е. нет состояний, находящихся в двух или более блоках. Состояния внутри блока взаимно эквивалентны. Это означает, что если p и r принадлежат блок состояний, эквивалентных q, то по теореме 2.2 они эквивалентны.

Допустим, есть два пересекающихся, но не совпадающих блока, скажем, в блоке B состояния p и q, а в блоке C есть состояние p, но нет q. По причине принадлежности p и q одном блоку, очевидна их эквивалентность.

Рассмотрим варианты построения блока C. Если он создавался состоянием p, то q также нужно было бы включить в этот блок, поскольку они эквивалентны. Значит, есть некое третье состояние s, порождающее блок C. Иными словами, C – множество состояний, которые эквивалентны s.

Состояния p и s эквивалентны, поскольку принадлежат C. Состояние q также эквивалентно p, ведь они принадлежат B. По теореме 2.2 состояние q эквивалентно s, но в таком случае q принадлежит блоку C. Это противоречит предположению о пересекающихся блоках. Таким образом, эквивалентность состояний задает их разбиение, т.е. всякие два состояния имеют совпадающие либо непересекающиеся множества эквивалентных состояний.

Теорема 2.3. Если для любого состояния ДКА создать блок, который состоит из этого состояния и эквивалентных ему, то различные блоки образуют **разбиение множества состояний**. Значит, любое состояние может принадлежать только одному блоку. Состояния в блоке эквивалентны, а в разных блоках – неэквивалентны.

Примем ее без доказательства. СЛАЙД 33.

Теперь мы можем смело перейти к формулированию алгоритма минимизации некоторого ДКА A. СЛАЙД 34.

(1) Для выявления всех пар эквивалентных состояний используем алгоритм заполнения таблицы.

(2) Для получения блоков взаимно эквивалентных состояний применяем к множеству Q алгоритм разбиения.

(3) Получаем ДКА B с минимальным числом состояний, используя результаты шага (2). Предположим, γ – функция переходов автомата B, S – множество эквивалентных состояний ДКА A, a – некоторый входной символ. Тогда должен существовать один блок состояний T, содержащий $\gamma(q, a)$ для всех состояний q из множества S. Если это не так, то a переводит состояния p и q из S в состояния из разных блоков согласно теореме 2.3. Отсюда, состояния p и q не эквивалентны и не могут принадлежать S. В результате определяем функцию переходов $\gamma(S, a) = T$.

Начальным состоянием ДКА B является блок с начальным состоянием ДКА A.

Множеством заключительных состояний ДКА B является множество блоков с заключительными состояниями ДКА A. Если одно состояние в блоке заключительное, то все остальные состояния в этом блоке тоже должны быть заключительными, поскольку всякое заключительное состояние различимо от всякого незаключительного, поэтому заключительное и незаключительное состояния не могут принадлежать одному блоку эквивалентных состояний.

Пример 12.

Минимизируем ДКА из примера 9. Эти состояния разбиваются на блоки следующим образом: $\{A, E\}$, $\{B, H\}$, $\{C\}$, $\{D, F\}$, $\{G\}$. Каждая пара эквивалентных состояний помещается в отдельный блок, а состояния, отличные от других, образуют отдельные блоки. В примере 11 мы определили блоки разбиения состояний. На СЛАЙДЕ 35 показан полученный МДКА. Пять его состояний соответствуют пяти блокам эквивалентных состояний оригинального ДКА.

Начальным состоянием МДКА является $\{A, E\}$, поскольку A начальное состояние неминимизированного ДКА. Единственным заключительным состоянием является $\{C\}$, т.к. C – это единственное заключительное состояние в ДКА. Переход из $\{A, E\}$ в $\{B, H\}$ по символу 0 очевиден, т.к. есть в оригинальном ДКА переход из A в B, а из E – в H. Все остальные переходы так же очевидны.

Литература к лекции 2

1. Гилл, А. Введение в теорию конечных автоматов / А. Гилл. – М.: Наука, 1966. – 272 с.
2. Кузнецов, А.С. Теория вычислительных процессов [Текст] : учеб. пособие / А. С. Кузнецов, М. А. Русаков, Р. Ю. Царев ; Сиб. федерал. ун-т. - Красноярск: ИПК СФУ, 2008. – 184 с.
3. Короткова, М.А. Математическая теория автоматов. Учебное пособие / М.А. Короткова. – М.: МИФИ, 2008. – 116 с.
4. Молчанов, А. Ю. Системное программное обеспечение. 3-е изд. / А.Ю. Молчанов. – СПб.: Питер, 2010. – 400 с.
5. Пример реализации конечных автоматов на языке C++ - http://www.devexp.ru/2011/02/konechnye-avtomaty-v-c/
6. Теория автоматов / Э. А. Якубайтис, В. О. Васюкевич, А. Ю. Гобземис, Н. Е. Зазнова, А. А. Курмит, А. А. Лоренц, А. Ф. Петренко, В. П. Чапенко // Теория вероятностей. Математическая статистика. Теоретическая кибернетика. — М.: ВИНИТИ, 1976. — Т. 13. — С. 109–188. — URL http://www.mathnet.ru/php/getFT.phtml?jrnid=intv&paperid=28&what=fullt&option_lang=rus
7. Серебряков В. А., Галочкин М. П., Гончар Д. Р., Фуругян М. Г. Теория и реализация

языков программирования — М.: МЗ-Пресс, 2006 г., 2-е изд. - http://trpl7.ru/t-books/TRYAP_BOOK_Details.htm

8. Finite State Machine Generator - http://sourceforge.net/projects/genfsm/
9. Введение в схемы, автоматы и алгоритмы - http://www.intuit.ru/studies/courses/1030/205/info