Введение в теорию автоматов, языков и вычислений

2-Е ИЗДАНИЕ

Introduction to Automata Theory, Languages, and Computation

SECOND EDITION

JOHN E. HOPCROFT
Cornell University

RAJEEV MOTWANI
Stanford University

JEFFREY D. ULLMAN
Stanford University

ADDISON-WESLEY PUBLISHING COMPANY

Boston • San Francisco • New York
London • Toronto • Sydney • Tokyo • Singapore • Madrid
Mexico City • Munich • Paris • Cape Town • Hong Kong • Montreal

Введение в теорию автоматов, языков и вычислений

2-Е ИЗДАНИЕ

ДЖОН ХОПКРОФТ
РАДЖИВ МОТВАНИ
ДЖЕФФРИ УЛЬМАН

Москва • Санкт-Петербург • Киев
2008

ББК 32.973.26-018.2.75
Х78
УДК 681.3.07

Издательский дом "Вильямс"

Перевод с английского *О.И. Васылык*, *М. Саит-Аметова*,
канд.физ.-мат.наук *А.Б. Ставровского*

Под редакцией канд.физ.-мат.наук *А.Б. Ставровского*

По общим вопросам обращайтесь в Издательский дом "Вильямс"
по адресу: info@williamspublishing.com, http://www.williamspublishing.com

Хопкрофт, Джон, Э., **Мотвани**, Раджив, **Ульман**, Джеффри, Д..

Х78 Введение в теорию автоматов, языков и вычислений, 2-е изд.. : Пер. с англ. — М. : Издательский дом "Вильямс", 2008. — 528 с. : ил. — Парал. тит. англ.

ISBN 978-5-8459-1347-0 (рус.)

Книга известных американских ученых посвящена теории автоматов и соответствующих формальных языков и грамматик - как регулярных, так и контекстно-свободных. Во второй части рассматриваются различные машины Тьюринга, при помощи которых формализуются понятия разрешимых и неразрешимых проблем, а также определяются функции временной и емкостной оценки сложности алгоритмов. Изложение ведется строго, но доступно, и сопровождается многочисленными примерами, а также задачами для самостоятельного решения.

Книга будет полезна читателям различных категорий - студентам, аспирантам, научным сотрудникам, преподавателям высших учебных заведений, а также всем, кто интересуется математическими основами современной вычислительной техники.

ББК 32.973.26-018.2.75

ISBN 978-5-8459-1347-0 (рус.)
ISBN 0-201-44124-1 (англ.)

Оглавление

Содержание

Предисловие

В предисловии к своей книге 1979 года, предшествовавшей данному изданию, Дж. Хопкрофт и Дж. Ульман с удивлением отмечали, что за время, прошедшее после выхода их первой книги в 1969 году, произошел взрыв в развитии теории автоматов. Действительно, книга, вышедшая в 1979 году, содержала множество тем, не затронутых в предыдущей работе, и по объему была почти вдвое больше. Сравнив эту книгу с книгой 1979 года, вы увидите, что она, как автомобили 1970-х "больше снаружи, но меньше изнутри". И хотя это может показаться шагом назад, мы считаем такие изменения целесообразными в силу ряда причин.

Во-первых, в 1979 году теория автоматов и языков еще активно развивалась, и одной из целей той книги было пробудить интерес математически одаренных студентов к исследованиям в этой области. На сегодня в теории автоматов имеется лишь узкое направление для исследований (чего не скажешь о ее приложениях). Поэтому, на наш взгляд, не имеет смысла сохранять здесь лаконичный, сугубо математический стиль книги 1979 года.

Во-вторых, за последние двадцать лет изменилась роль теории автоматов и языков. В 1979 году это был сложный предмет, требующий от читателя высокого уровня подготовки. Читатель нашей книги, в особенности последних ее глав, представлялся нам хорошо подготовленным студентом-старшекурсником. Сегодня же этот предмет входит в стандартную вузовскую программу для младших курсов. Поэтому содержание книги должно быть по возможности доступным, а следовательно, содержать больше подробных доказательств и обоснований по сравнению с предыдущей книгой.

В-третьих, за два последних десятилетия информатика (Computer Science) невообразимо разрослась. И если в 1979 году вузовскую программу приходилось искусственно заполнять предметами, которые, как нам казалось, могли послужить новой волне развития технологий, то сегодня таких дисциплин уже слишком много.

В-четвертых, за эти годы информатика стала практическим предметом, и многие изучающие ее студенты настроены прагматически. Мы по-прежнему убеждены, что теория автоматов является мощным инструментом для исследований во множестве новых областей. А упражнения теоретического, развивающего плана, которые содержит обычный курс теории автоматов, сохраняют свое значение вне зависимости от того, предпочитает ли студент изучать лишь практические, "денежные" технологии. Однако для того, чтобы обеспечить данному предмету достойное место среди прочих дисциплин, изучаемых студентами-информатиками, нам представляется необходимым, наряду с математической частью, подчеркнуть и практические приложения теории. С этой целью мы заменили часть довольно сложных для понимания тем из предыдущей книги актуальными примерами применения этих идей на практике. В то время как приложения теории автоматов

и языков для компиляторов сегодня настолько понятны и естественны, что входят во всякий стандартный курс по компиляторам, существует немало и совершенно новых приложений. Например, алгоритмы проверки моделей, используемые для верификации протоколов, и языки описания документов, построенные по образцу контекстно-свободных грамматик.

И, наконец, последнее объяснение одновременного увеличения объема книги и уменьшения ее фактического содержания состоит в том, что при ее верстке использовались все преимущества издательских систем TEX и LATEX, которые поощряют "открытый" стиль печати, благодаря чему книга увеличивается в объеме, но становится удобнее для чтения. Мы с благодарностью отмечаем труд создателей этих издательских систем Дона Кнута и Леса Лампорта.

Как пользоваться книгой

Эта книга рассчитана на полусеместровый или семестровый курс лекций для студентов второго года обучения и выше. На ее основе в Стэндфордском университете Раджи-вом и Джеффом читается полусеместровый курс (CS154) по теории автоматов и языков. Ввиду ограниченности по времени в этом курсе не охвачены глава 11 и часть материала главы 10, например, довольно сложные вопросы о полиномиально-временно*й* сводимости. На Web-сайте книги (см. ниже) помещено несколько сокращенных вариантов этого курса с замечаниями.

Несколько лет назад мы столкнулись с тем, что многие студенты, поступившие в Стэнфорд после окончания колледжа, прошли курс теории автоматов, не содержавший теорию сложности. Преподавательский состав Стэнфорда полагает, что для всякого, кто серьезно занимается информатикой, эти идеи чрезвычайно важны. Тут недостаточно знать лишь то, что "для решения NP-полной задачи требуется очень много времени". Специально для таких студентов был разработан курс лекций (CS154N), который содержит материал только 8, 9 и 10 глав. Фактически, для того, чтобы освоить курс CS154N, они прослушивают лишь последнюю треть курса CS154. И по сегодняшний день на каждом курсе находится несколько студентов, желающих заниматься именно таким образом. Мы рекомендуем этот подход, поскольку он не требует чрезмерных усилий.

Требования к уровню подготовки

Чтение этой книги не вызовет затруднений у студентов, освоивших основы дискретной математики, в том числе изучивших графы, деревья, логику и методы доказательств. Кроме того, мы предполагаем, что читатель в достаточной степени знаком с программированием и, в частности, имеет представление об общих структурах данных, рекурсии и роли таких главных системных компонентов, как компиляторы. Эта сумма знаний соответствует стандартной программе первых двух лет обучения для студентов, изучающих информатику.

Упражнения

Книга содержит большое число упражнений, по несколько почти в каждом разделе. Упражнения или части упражнений повышенной сложности отмечены восклицательным знаком. Наиболее трудные из них отмечены двумя восклицательными знаками.

Некоторые упражнения отмечены звездочкой. Их решения мы намерены поместить на Web-странице книги. Они предназначены для самопроверки и доступны широкой аудитории. Отметим, что в некоторых случаях в одном упражнении *Б* требуется определенным образом изменить или адаптировать ваше решение другого упражнения *А*. Если какая-то часть упражнения *А* имеет решение, то естественно ожидать, что соответствующая часть упражнения *Б* также имеет решение.

Поддержка в World Wide Web

Адрес книги в Internet:

`http://www-db.stanford.edu/~ullman/ialc.html`

Здесь вы найдете решения заданий, отмеченных звездочкой, список замеченных опечаток и некоторые вспомогательные материалы. По мере чтения лекций по курсу CS154 мы постараемся размещать тут наши замечания по поводу домашних заданий, решений упражнений и экзаменов.

Благодарности

На часть материала главы 1 повлияла рукопись Крейга Сильверштейна (Creig Silverstein) о том, “как строить доказательства”. Мы благодарим также следующих лиц: Зое Абрамс (Zoe Abrams), Джордж Кандеа (George Candea), Хаовен Чен (Haowen Chen), Бьен-Ган Чан (Byong-Gun Chun), Джеффри Шаллит (Jeffrey Shallit), Брет Тейлор (Bret Taylor), Джейсон Таунсенд (Jason Townsend) и Эрик Узурео (Erik Uzureau), которые высказали свои замечания и указали на ряд опечаток в черновом варианте книги. Ошибки, оставшиеся незамеченными, авторы безусловно относят на свой счет.

Джон Хопкрофт
Раджив Мотвани
Джефри Ульман
Итака, Нью-Йорк и Стэнфорд, Калифорния
Сентябрь, 2000.

ГЛАВА 1

Автоматы: методы и понятия

Теория автоматов занимается изучением абстрактных вычислительных устройств, или "машин". В 1930-е годы, задолго до появления компьютеров, А. Тьюринг исследовал абстрактную машину, которая, по крайней мере в области вычислений, обладала всеми возможностями современных вычислительных машин. Целью Тьюринга было точно описать границу между тем, что вычислительная машина может делать, и тем, чего она не может. Полученные им результаты применимы не только к абстрактным *машинам Тьюринга*, но и к реальным современным компьютерам.

В 1940-х и 1950-х годах немало исследователей занимались изучением простейших машин, которые сегодня называются "конечными автоматами". Такие автоматы вначале были предложены в качестве модели функционирования человеческого мозга. Однако вскоре они оказались весьма полезными для множества других целей, которые будут упомянуты в разделе 1.1. А в конце 1950-х лингвист Н. Хомский занялся изучением формальных "грамматик". Не будучи машинами в точном смысле слова, грамматики, тем не менее, тесно связаны с абстрактными автоматами и служат основой некоторых важнейших составляющих программного обеспечения, в частности, компонентов компиляторов.

В 1969 году С. Кук развил результаты Тьюринга о вычислимости и невычислимости. Ему удалось разделить задачи на те, которые могут быть эффективно решены вычислительной машиной, и те, которые, в принципе, могут быть решены, но требуют для этого так много машинного времени, что компьютер оказывается бесполезным для решения практически всех экземпляров задачи, за исключением небольших. Задачи последнего класса называют "трудно разрешимыми" ("труднорешаемыми") или "NP-трудными". Даже при экспоненциальном росте быстродействия вычислительных машин ("закон Мура") весьма маловероятно, что нам удастся достигнуть значительных успехов в решении задач этого класса.

Все эти теоретические построения непосредственно связаны с тем, чем занимаются ученые в области информатики сегодня. Некоторые из введенных понятий, такие, например, как конечные автоматы и некоторые типы формальных грамматик, используются при проектировании и создании важных компонентов программного обеспечения. Другие понятия, например, машина Тьюринга, помогают нам уяснить принципиальные возможности программного обеспечения. В частности, теория сложности вычислений

позволяет определить, можем ли мы решить ту или иную задачу “в лоб” и написать соответствующую программу для ее решения (если эта задача не принадлежит классу “трудно разрешимых”), или же нам следует искать решение данной трудно разрешимой задачи в обход, используя приближенный, эвристический, или какой-либо другой метод, с помощью которого удастся ограничить время, затрачиваемое программой на ее решение.

В этой вводной главе дается обзор содержания теории автоматов и ее приложений. Основная часть главы посвящена рассмотрению различных методов доказательств и способов их нахождения. Рассматриваются дедуктивные доказательства, утверждения, получаемые переформулировкой, доказательства от противного, доказательства по индукции и другие важные понятия. В последней части вводится ряд основополагающих понятий теории автоматов: алфавиты, цепочки и языки.

1.1. Зачем изучается теория автоматов?

В силу ряда причин теория автоматов и теория сложности являются важнейшей частью ядра информатики. В этом разделе мы познакомим читателя с главными мотивами, побуждающими нас к их изучению, а также обрисуем в общих чертах основные темы, затрагиваемые в книге.

1.1.1. Введение в теорию конечных автоматов

Конечные автоматы являются моделью для многих компонентов аппаратного и программного обеспечения. Ниже (начиная со второй главы) мы рассмотрим примеры их использования, а сейчас просто перечислим наиболее важные из них.

1. Программное обеспечение, используемое для разработки и проверки цифровых схем.
2. “Лексический анализатор” стандартного компилятора, т.е. тот компонент компилятора, который отвечает за разбивку исходного текста на такие логические единицы, как идентификаторы, ключевые слова и знаки пунктуации.
3. Программное обеспечение для сканирования таких больших текстовых массивов, как наборы Web-страниц, с целью поиска заданных слов, фраз или других последовательностей символов (шаблонов).
4. Программное обеспечение для проверки различного рода систем (протоколы связи или протоколы для защищенного обмена информацией), которые могут находиться в конечном числе различных состояний.

С точными определениями автоматов различного типа мы встретимся вскоре. А начнем с краткого описания того, что представляет собой конечный автомат, и как он действует. Существует множество различных систем и их компонентов, например,

только что перечисленные, которые можно рассматривать, как находящиеся в каждый момент времени в одном из конечного числа "состояний". Назначением каждого состояния является запоминание определенного момента истории системы. Поскольку число состояний конечно, то запомнить всю историю системы невозможно, а значит, необходимо строить систему тщательно, чтобы хранить только действительно важную информацию и забывать несущественную. Преимущество конечности числа состояний заключается в том, что систему можно реализовать, имея ограниченные ресурсы. Например, ее можно реализовать "в железе" как схему или же в виде простой программы, принимающей решения на основе ограниченного количества данных или текущей команды машинного кода.

Пример 1.1. Простейшим нетривиальным конечным автоматом является переключатель "включено-выключено". Это устройство помнит свое текущее состояние, и от этого состояния зависит результат нажатия кнопки. Из состояния "выключено" нажатие кнопки переводит переключатель в состояние "включено", и наоборот.

На рис. 1.1 представлена конечноавтоматная модель переключателя. Как и для всех конечных автоматов, состояния обозначены кружками. В данном случае они отмечены как "*вкл*." и "*выкл*.". Дуги между состояниями отмечены "входными символами", задающими внешние воздействия на систему. В нашем случае это *Нажатие*, что означает нажатие на кнопку переключателя. Стрелки указывают, что всякий раз при *Нажатии* система переходит из одного состояния в другое.

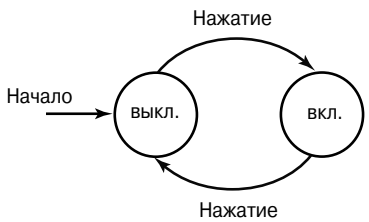

Рис. 1.1. Конечный автомат, моделирующий переключатель

Одно из состояний является так называемым "начальным состоянием". Это состояние (в нашем случае — "*выкл*."), в котором система находится изначально. На рисунке оно отмечено словом *Начало* и стрелкой, указывающей на это состояние.

Часто необходимо выделять одно или несколько "заключительных", или "допускающих", состояний. Попав в одно из них в результате реализации некоторой последовательности входных воздействий, мы можем считать такую входную последовательность в определенном смысле "хорошей". Так, например, состояние "*вкл*." на рис. 1.1 можно рассматривать как допускающее, поскольку если переключатель находится в этом состоянии, то устройство, управляемое им, находится в рабочем режиме. Допускающие состояния принято обозначать двойным кружком, хотя мы не использовали это обозначение на рис 1.1. □

Пример 1.2. Иногда состояние автомата запоминает гораздо более сложную информацию, чем выбор "*вкл.–выкл.*". На рис. 1.2 представлен конечный автомат, который может служить частью лексического анализатора. Его функцией является распознавание ключевого слова `then`. Этот автомат должен иметь пять различных состояний, каждое из которых представляет позицию в слове `then`, достигнутую на данный момент. Эти позиции соответствуют префиксам слова, от пустой цепочки (никакая позиция в слове еще не достигнута) до целого слова.

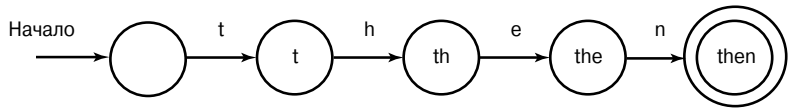

Рис. 1.2. Конечный автомат, моделирующий распознавание слова `then`

На рис. 1.2 каждое из пяти состояний обозначено частью слова, прочитанной на данном шаге. Входным сигналам соответствуют буквы. Мы можем считать, что данный лексический анализатор всякий раз просматривает по одному символу компилируемой программы. Каждый следующий символ рассматривается как входной сигнал для данного автомата. Начальное состояние автомата соответствует пустой цепочке, и каждое состояние имеет переход по очередной букве слова `then` в состояние, соответствующее следующему префиксу. Состояние, обозначенное словом "then", достигается, когда по буквам введено все данное слово. Поскольку функция автомата заключается в распознавании слова `then`, то последнее состояние будем считать единственным допускающим. □

1.1.2. Структурные представления

Следующие системы записи не являются автоматными, но играют важную роль в теории автоматов и ее приложениях.

1. *Грамматики*. Они являются полезными моделями при проектировании программного обеспечения, обрабатывающего данные рекурсивной структуры. Наиболее известный пример — "синтаксический анализатор". Этот компонент компилятора работает с такими рекурсивно вложенными конструкциями в типичных языках программирования, как выражения: арифметические, условные и т.п. Например, грамматическое правило типа $E \Rightarrow E + E$ означает, что выражение может быть получено соединением любых двух выражений с помощью знака "плюс". Это типичное правило построения выражений в существующих языках программирования. Ниже, в главе 5, будут определены так называемые контекстно-свободные грамматики.
2. *Регулярные выражения*. Они также задают структуру данных, в частности, текстовых цепочек. Как будет показано в главе 3, шаблоны описываемых ими цепочек представляют собой то же самое, что задают конечные автоматы. Стиль этих выражений суще-