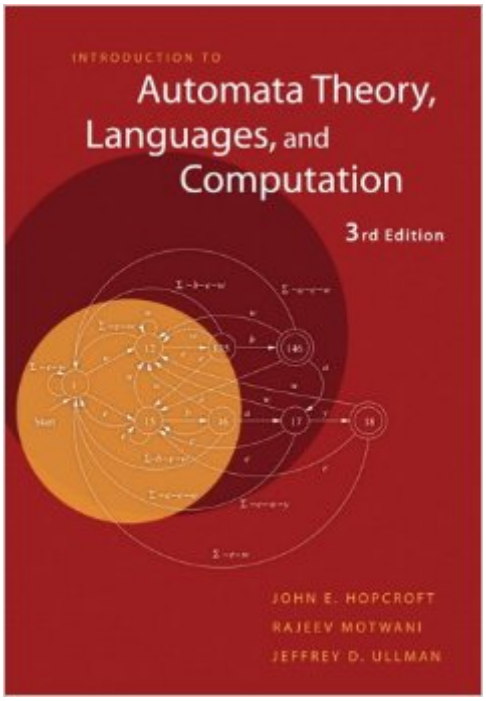
Introduction To Automata Theory, Languages, And Computation (3rd Edition)



Synopsis

This classic book on formal languages, automata theory, and computational complexity has been updated to present theoretical concepts in a concise and straightforward manner with the increase of hands-on, practical applications. This new edition comes with Gradiance, an online assessment tool developed for computer science. Gradiance is the most advanced online assessment tool developed for the computer science discipline. With its innovative underlying technology, Gradiance turns basic homework assignments and programming labs into an interactive learning experience for students. By using a series of root questions and hints, it not only tests a student's capability, but actually simulates a one-on-one teacher-student tutorial that allows for the student to more easily learn the material. Through the programming labs, instructors are capable of testing, tracking, and honing their students' skills, both in terms of syntax and semantics, with an unprecedented level of assessment never before offered. For more information about Gradiance, please visit www.aw.com/gradiance.

Book Information

Hardcover: 750 pages

Publisher: Pearson; 3 edition (July 9, 2006)

Language: English

ISBN-10: 0321455363

ISBN-13: 978-0321455369

Product Dimensions: 6.6 x 1.4 x 9.4 inches

Shipping Weight: 2 pounds (View shipping rates and policies)

Average Customer Review: 3.5 out of 5 starsÂ Â See all reviewsÂ (55 customer reviews)

Best Sellers Rank: #98,526 in Books (See Top 100 in Books) #16 inÂ Books > Computers & Technology > Computer Science > AI & Machine Learning > Machine Theory #28 inÂ Books > Science & Math > Mathematics > Pure Mathematics > Discrete Mathematics #59 inÂ Books > Science & Math > Mathematics > Pure Mathematics > Logic

Customer Reviews

This is a good book - but as a revision of a much-revered classic of the field, it's a bit of a disappointment. Hopcroft & Ullman wrote the classic text way back in 1969, and then revised it in 1979. It was pretty much the standard text the world over for an introduction to the theory of computation. But over the last two decades, more and more people have been studying Computer science, and many of them have no time for theory and formalism and all the 'dry stuff' ..... The

authors point out that because of such reasons and also because nowadays there's little research in the theory of computation per se, and more in its applications, they've written a book to cater to today's students. Which, in other words, means they've simplified the presentation, tried to provide intuition whenever possible, given lots more examples and done away with some of the more difficult material. This approach puts the book into direct competition with Michael Sipser's excellent 'Introduction to the theory of computation', a contest it cannot win, though it might be a respectable second. Almost all topics are motivated by giving examples of how they're related to applications in the 'real world', and similar to Sipser's 'proof idea' approach, the authors first present a topic informally and then formally, thus gently leading the reader to the formal proofs.

I've just passed my exam on Theory of Computation, and I've used both editions of this text. Frankly speaking, I couldn't choose one of the two should I keep only one of them.Whereas the first was full of strict formalism, the second has traded this for a more discursive approach. Whereas the first reported theorems name (of their authors), the second has traded this for a richer bibliography at the end of the chapters. And more objectively, the first edition covered more "classical" topics with shorter treatments than the second, but this last treats survived topics with richer details (starting from the first chapter on mathematical basis for the course) and with updated examples of applications (XML and Markup Languages, e-commerce for DFA, etc).This said, you know why I can't decide. A discursive approach is of course always desiderable, especially if you're completely new to a subject, but a strong notation is helpful in my mind because it improves communication and removes ambiguities. Hence, the best approach would probably have been a mix of the two, or halfway the two.As a second matter, having a rich bibliography is surely helpful both for further studies and as a reference, but it's quite tedious to look at the index and be unable to find something like "Kleene theorem": you've to dive into bibligraphy to discover that "L is an L(DFA) if and only if it also is L(REG)" is something that has been studied by Kleene.

Computation Formal Languages and Their Relation to Automata (Addison-Wesley Series in Computer Science and Information Processing) The Languages of Tolkien's Middle-Earth: A Complete Guide to All Fourteen of the Languages Tolkien Invented Introduction to the Theory of Computation Finite Automata and Regular Expressions: Problems and Solutions The Wonderland Of Music Boxes And Automata Amazing Automata -- Dinosaurs! (Dover Origami Papercraft) Paper Models That Rock!: Six Pendulum Automata (Dover Origami Papercraft) Introduction to Statistical Relational Learning (Adaptive Computation and Machine Learning series) Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning series) Introduction to Machine Learning (Adaptive Computation and Machine Learning series) Common LISP: A Gentle Introduction to Symbolic Computation (Dover Books on Engineering) Masterminds of Programming: Conversations with the Creators of Major Programming Languages (Theory in Practice (O'Reilly)) An Introduction to Music Therapy: Theory and Practice, 3rd Edition Implementing Programming Languages. an Introduction to Compilers and Interpreters (Texts in Computing)