



TEORI BAHASA dan Automata

Fondasi Formal untuk Komputasi

Widhiya Ninsiana, Dahlan Abdullah
Ni Ketut Widhiarcani Matradewi, Cut Ita Erliana
Zalfie Ardian

Teori Bahasa & **AUTOMATA**

**Widhiya Ninsiana
Dahlan Abdullah
Ni Ketut Widhiarcani Matradewi
Cut Ita Erliana
Zalfie Ardian**



Penerbit KBM Indonesia

Adalah penerbit dengan misi memudahkan proses penerbitan buku-buku penulis di tanah air indonesia, serta menjadi media *sharing* proses penerbitan buku

TEORI BAHASA DAN AUTOMATA

Copyright @2026 by Widhiya Ninsiana, dkk

All rights reserved

PENERBIT KARYA BAKTI MAKMUR (KBM) INDONESIA

Anggota IKAPI (Ikatan Penerbit Indonesia) **NO. IKAPI 279/JTI/2021**

Depok, Sleman-Jogjakarta (Kantor) **081357517526 (Tlpn/WA)**

Penulis

Widhiya Ninsiana

Dahlan Abdullah

Ni Ketut Widhiarcani Matradewi

Cut Ita Erliana

Zalfie Ardian

Desain Sampul

Eca Art

Tata Letak

Lailatul Nur Ula

Editor Naskah

Chicha Rizka Gunawan, S.Kom., M.Kom.

E-ISBN

978-634-278-375-7

15,5 x 23 cm, viii + 452 halaman

Cetakan ke-1, Mei 2026

Isi buku diluar tanggungjawab penerbit

Hak cipta merek KBM Indonesia sudah terdaftar di

DJKI-Kemenkumham dan isi buku dilindungi undang-undang

Dilarang keras menerjemahkan, memfotokopi, atau

Memperbanyak sebagian atau seluruh isi buku ini

Tanpa seizin penerbit karena beresiko sengketa hukum

Website

<https://penerbitkbm.com> | www.penerbitbukumurah.com

Instagram

@penerbit.kbmindonesia | @penerbitbukujogja

Email

naskah@penerbitkbm.com

Distributor

<https://penerbitkbm.com/toko-buku/>

Youtube

Penerbit KBM Sastrabook

Sanksi Pelanggaran Pasal 113

Undang-Undang No. 28 Tahun 2014 Tentang Hak Cipta

- (i) Setiap Orang yang dengan tanpa hak melakukan pelanggaran hak ekonomi sebagaimana dimaksud dalam Pasal 9 ayat (1) huruf i untuk Penggunaan Secara Komersial dipidana dengan pidana penjara paling lama 1 (satu) tahun dan/atau pidana denda paling banyak Rp 100.000.000 (seratus juta rupiah).
- (ii) Setiap Orang yang dengan tanpa hak dan/atau tanpa izin Pencipta atau pemegang Hak Cipta melakukan pelanggaran hak ekonomi Pencipta sebagaimana dimaksud dalam Pasal 9 ayat (1) huruf c, huruf d, huruf f, dan/atau huruf h untuk Penggunaan Secara Komersial dipidana dengan pidana penjara paling lama 3 (tiga) tahun dan/atau pidana denda paling banyak Rp 500.000.000,00 (lima ratus juta rupiah).
- (iii) Setiap Orang yang dengan tanpa hak dan/atau tanpa izin Pencipta atau pemegang Hak Cipta melakukan pelanggaran hak ekonomi Pencipta sebagaimana dimaksud dalam Pasal 9 ayat (1) huruf a, huruf b, huruf e, dan/atau huruf g untuk Penggunaan Secara Komersial dipidana dengan pidana penjara paling lama 4 (empat) tahun dan/atau pidana denda paling banyak Rp 1.000.000.000,00 (satu miliar rupiah).
- (iv) Setiap Orang yang memenuhi unsur sebagaimana dimaksud pada ayat (3) yang dilakukan dalam bentuk pembajakan, dipidana dengan pidana penjara paling lama 10 (sepuluh) tahun dan/atau pidana denda paling banyak Rp 4.000.000.000,00 (empat miliar rupiah).

PRAKATA

Penulisan buku *Teori Bahasa dan Automata* ini dilatarbelakangi oleh kebutuhan akan sumber referensi komprehensif yang menjembatani teori fundamental dengan aplikasi praktis dalam ilmu komputer. Tujuannya adalah menyajikan konsep-konsep inti Teori Bahasa dan Automata secara sistematis dan mudah dipahami, mulai dari dasar hingga topik lanjutan. Buku ini mencakup ruang lingkup yang luas, mulai dari bahasa formal, tata bahasa, hirarki Chomsky, berbagai jenis automata, hingga komputabilitas dan kompleksitas. Ditujukan bagi mahasiswa ilmu komputer, informatika, rekayasa perangkat lunak, serta para praktisi dan peneliti yang ingin memperdalam pemahaman tentang fondasi komputasi.

Kami berharap buku ini dapat menjadi panduan yang berharga, tidak hanya untuk memahami teori di balik komputasi, tetapi juga untuk menginspirasi pembaca dalam mengembangkan solusi inovatif di era digital. Ucapan terima kasih tak terhingga kami sampaikan kepada semua pihak yang telah berkontribusi dalam penyusunan buku ini, baik secara langsung maupun tidak langsung. Semoga buku ini dapat memberikan kontribusi nyata bagi perkembangan ilmu pengetahuan dan teknologi.

Penulis

KATA PENGANTAR

Buku *Teori Bahasa dan Automata* ini hadir sebagai kontribusi signifikan dalam literatur ilmu komputer, menyajikan eksplorasi mendalam tentang pilar-pilar teoretis yang menopang dunia komputasi modern. Sebagai editor, kami merasa terhormat dapat menyajikan karya ini, yang dengan cermat menguraikan kompleksitas bahasa formal, automata, dan teori komputabilitas, menjadikannya dapat diakses oleh khalayak luas.

Struktur buku ini dirancang secara logis dan progresif, dimulai dengan pengantar yang komprehensif mengenai ruang lingkup dan sejarah Teori Bahasa dan Automata, kemudian secara bertahap memperkenalkan konsep-konsep dasar bahasa formal, tata bahasa, dan hirarki Chomsky. Pembahasan dilanjutkan dengan detail mengenai berbagai jenis automata, seperti *Finite Automata*, *Pushdown Automata*, hingga *Mesin Turing*, yang merupakan inti dari pemahaman komputasi. Bagian akhir buku ini secara apik mengaitkan teori-teori tersebut dengan isu-isu krusial seperti komputabilitas, kompleksitas, dan aplikasi praktis dalam teknologi modern, termasuk desain bahasa pemrograman, kompilator, kecerdasan buatan, dan pemrosesan bahasa alami.

Kami yakin bahwa buku ini akan menjadi referensi yang tak ternilai bagi mahasiswa, akademisi, dan profesional di bidang ilmu komputer. Kedalaman materi, kejelasan penjelasan, dan relevansi aplikasi yang disajikan menjadikan buku ini esensial bagi siapa pun yang ingin memahami fondasi teoretis di balik setiap interaksi kita dengan teknologi digital. Penulis telah berhasil menciptakan sebuah karya yang tidak hanya informatif tetapi juga inspiratif, membuka wawasan tentang bagaimana prinsip-prinsip abstrak ini membentuk dunia komputasi yang kita kenal.

DAFTAR ISI

PRAKATA	i
KATA PENGANTAR	iii
DAFTAR ISI	v

BAB 1

PENDAHULUAN TEORI BAHASA DAN AUTOMATA.....	1
1.1. Ruang Lingkup Teori Bahasa dan Automata.....	1
1.2. Peran Bahasa Formal dalam Ilmu Komputer.....	7
1.3. Sejarah dan Perkembangan Teori Automata	12
1.4. Hubungan Teori Bahasa dengan Algoritma dan Komputasi.....	17
1.5. Peta Konsep dan Struktur Pembahasan Buku.....	23

BAB 2

KONSEP DASAR BAHASA FORMAL	29
2.1. Definisi Alfabet, String, dan Bahasa	29
2.2. Operasi - Operasi pada Bahasa Formal	34
2.3. Notasi dan Representasi Bahasa	42
2.4. Klasifikasi Bahasa Formal.....	47
2.5. Bahasa Formal dalam Pemodelan Sistem	53

BAB 3

GRAMMAR DAN PEMBENTUKAN BAHASA	59
3.1. Konsep Grammar Formal.....	59
3.2. Komponen Grammar dan Aturan Produksi.....	65
3.3. Derivasi dan Pohon Parse.....	72
3.4. Ambiguitas Grammar	74
3.5. Peran Grammar dalam Kompilator	76

BAB 4

HIRARKI CHOMSKY	83
4.1. Pengantar Hirarki Chomsky	83
4.2. Grammar Tipe-3 (<i>Regular Grammar</i>)	90
4.3. Grammar Tipe-2 (<i>Context-Free Grammar</i>)	97
4.4. Grammar Tipe-1 (<i>Context-Sensitive Grammar</i>) dan Tipe-0 (<i>Unrestricted Grammar</i>).....	103
4.5. Implikasi Hirarki Chomsky dalam Komputasi	108

BAB 5

AUTOMATA HINGGA (FINITE AUTOMATA)	115
5.1. Konsep dan Definisi Finite Automata	115
5.2. <i>Deterministic Finite Automata</i> (DFA)	121
5.3. <i>Non-Deterministic Finite Automata</i> (NFA)	127
5.4. Ekuivalensi DFA dan NFA.....	135
5.5. Aplikasi Finite Automata.....	143

BAB 6

EKSPRESI REGULAR DAN BAHASA REGULAR.....	151
6.1. Definisi Ekspresi Regular.....	151
6.2. Hubungan Ekspresi Regular dan Automata	159
6.3. Konversi Ekspresi Regular ke Automata.....	165
6.4. Sifat-sifat Bahasa Regular	173
6.5. Implementasi Bahasa Regular dalam Sistem Nyata.....	180

BAB 7

SIFAT PENUTUPAN DAN KEPUTUSAN BAHASA REGULAR ...	187
7.1. Operasi Penutupan Bahasa Regular	187
7.2. Teorema Penutupan Bahasa Regular.....	194
7.3. Pumping Lemma untuk Bahasa Regular.....	202
7.4. Keputusan dan Ketakterputusan.....	209
7.5. Analisis Keterbatasan Bahasa Regular	215

BAB 8

PUSHDOWN AUTOMATA (PDA)	221
8.1. Konsep Dasar Pushdown Automata	221
8.2. Struktur dan Mekanisme Stack.....	228

8.3. <i>Deterministic dan Non-Deterministic PDA</i>	234
8.4. Hubungan PDA dan <i>Context-Free Grammar</i>	241
8.5. Aplikasi PDA dalam Parsing.....	249

BAB 9

BAHASA BEBAS KONTEKS (*CONTEXT-FREE LANGUAGES*).....257

9.1. Karakteristik Bahasa Bebas Konteks	257
9.2. Normal Form Grammar Bebas Konteks	263
9.3. Ambiguitas dan Eliminasi Ambiguitas.....	270
9.4. Pumping Lemma untuk Bahasa Bebas Konteks	271
9.5. Batasan Bahasa Bebas Konteks	274

BAB 10

MESIN TURING 281

10.1. Konsep Dasar Mesin Turing.....	281
10.2. Komponen dan Cara Kerja Mesin Turing	287
10.3. Mesin Turing Deterministik dan Non-Deterministik	295
10.4. Bahasa yang Dikenali Mesin Turing	302
10.5. Mesin Turing sebagai Model Komputasi Universal.....	305

BAB 11

KOMPUTABILITAS DAN MASALAH KEPUTUSAN 313

11.1. Konsep Komputabilitas	313
11.2. Masalah Keputusan dalam Komputasi	319
11.3. Bahasa Rekursif dan Rekursif Terhitung.....	325
11.4. Masalah yang Tidak Dapat Diputuskan.....	328
11.5. Implikasi Teori Komputabilitas.....	335

BAB 12

KOMPLEKSITAS WAKTU DAN RUANG 341

12.1. Pengantar Teori Kompleksitas	341
12.2. Kelas Kompleksitas Waktu.....	347
12.3. Kelas Kompleksitas Ruang	353
12.4. Masalah P, NP, dan NP-Complete.....	360
12.5. Dampak Kompleksitas terhadap Desain Algoritma	367

BAB 13

AUTOMATA DAN BAHASA DALAM TEKNOLOGI MODERN 373

- 13.1. Automata dalam Desain Bahasa Pemrograman 373
- 13.2. Teori Bahasa dalam Compiler dan Interpreter 379
- 13.3. Automata dalam Sistem Keamanan dan Verifikasi.....383
- 13.4. Penerapan dalam Artificial Intelligence.....388
- 13.5. Automata dalam Pemrosesan Bahasa Alami.....393

BAB 14

TREN TERKINI DAN PENGEMBANGAN TEORI BAHASA DAN

AUTOMATA 401

- 14.1. Automata dan Formal Methods.....401
- 14.2. Model Komputasi Baru dan Extended Automata.....407
- 14.3. Teori Bahasa dalam *Big Data* dan *Machine Learning* 412
- 14.4. Tantangan dan Arah Riset Masa Depan 417
- 14.5. Peran Teori Bahasa dan Automata di Era Digital.....423

GLOSARIUM.....429

REFERENSI.....440

PROFIL PENULIS.....452

BAB 1

PENDAHULUAN TEORI BAHASA DAN AUTOMATA

1.1. Ruang Lingkup Teori Bahasa dan Automata

Teori Bahasa dan Automata (TBA) merupakan salah satu fondasi fundamental dalam ilmu komputer teoretis, yang menyediakan kerangka kerja formal untuk memahami batasan dan kemampuan komputasi. Bidang ini secara esensial mengeksplorasi hubungan antara bahasa formal—kumpulan string yang dibangun berdasarkan aturan tertentu—dan automata—model matematis dari mesin komputasi abstrak yang mampu mengenali atau menghasilkan bahasa-bahasa tersebut. Ruang lingkup TBA sangat luas, mencakup spektrum dari konsep-konsep abstrak hingga aplikasi praktis yang mendalam dalam berbagai disiplin ilmu komputasi modern.

1.1.1 Definisi dan Konsep Dasar

Pada intinya, TBA berpusat pada dua entitas utama: *bahasa formal* dan *automata*. Bahasa formal adalah himpunan string yang dibentuk dari alfabet tertentu, mengikuti serangkaian aturan tata bahasa yang presisi. Berbeda dengan bahasa alami yang ambigu dan kontekstual, bahasa formal didefinisikan secara matematis, memungkinkan analisis yang ketat. Contoh sederhana adalah bahasa yang terdiri dari semua string biner yang dimulai dengan '0' dan diakhiri dengan '1'. Aturan pembentukannya jelas, dan setiap string yang memenuhi kriteria tersebut adalah bagian dari bahasa.

Automata, di sisi lain, adalah model matematis dari mesin abstrak yang dirancang untuk memproses bahasa formal. Automata dapat dianggap sebagai pengenalan bahasa, yang menentukan apakah suatu string masukan termasuk dalam bahasa tertentu atau tidak. Model automata bervariasi dalam kompleksitas dan kemampuan komputasinya, mulai dari *finite automata* (FA) yang paling sederhana hingga *Turing machine* (TM) yang paling kuat. Setiap jenis automata memiliki kemampuan komputasi yang berbeda, yang secara langsung berkorelasi dengan kelas bahasa formal yang dapat dikenali atau dihasilkan olehnya.

Hubungan antara bahasa formal dan automata adalah inti dari TBA. *Hierarki Chomsky*, yang diperkenalkan oleh Noam Chomsky, mengklasifikasikan bahasa formal ke dalam empat tingkatan berdasarkan kompleksitas tata bahasanya dan jenis automata yang dapat mengenalinya (Chomsky, 1956). Hierarki ini mencakup:

- **Bahasa Regular:** Dikenali oleh *Finite Automata* (FA). Ini adalah bahasa paling sederhana, sering digunakan dalam ekspresi reguler, analisis leksikal kompilator, dan desain sirkuit digital.
- **Bahasa Bebas Konteks:** Dikenali oleh *Pushdown Automata* (PDA). Bahasa ini lebih kompleks dan banyak digunakan dalam sintaksis bahasa pemrograman, parsing, dan representasi struktur data hierarkis.
- **Bahasa Sensitif Konteks:** Dikenali oleh *Linear Bounded Automata* (LBA). Bahasa ini memiliki aturan tata bahasa yang lebih ketat, di mana produksi bergantung pada konteks simbol.
- **Bahasa Rekursif Enumerable:** Dikenali oleh *Turing Machine* (TM). Ini adalah kelas bahasa paling umum dan mencakup semua bahasa yang dapat diproses oleh algoritma komputasi apa pun.

Pemahaman tentang hierarki ini sangat penting karena memberikan wawasan tentang batasan komputasi. Tidak semua masalah dapat diselesaikan oleh setiap jenis automata, dan pemahaman tentang batasan ini adalah kunci dalam merancang algoritma dan sistem komputasi yang efisien.

1.1.2 Aplikasi dalam Ilmu Komputer

Meskipun TBA sering dianggap sebagai disiplin ilmu teoretis, aplikasinya meresap ke hampir setiap aspek ilmu komputer modern. Pemahaman yang kuat tentang TBA sangat penting bagi para ilmuwan komputer dan insinyur perangkat lunak.

1.1.2.1 Kompilator dan Interpreter

Salah satu aplikasi paling klasik dan signifikan dari TBA adalah dalam desain *kompilator* dan *interpreter*. Proses kompilasi, yang mengubah kode sumber bahasa tingkat tinggi menjadi kode mesin, secara fundamental bergantung pada prinsip-prinsip TBA.

- **Analisis Leksikal:** Tahap pertama dalam kompilasi adalah analisis leksikal, di mana kode sumber dipecah menjadi unit-unit dasar yang disebut *token*. Proses ini secara efektif diimplementasikan menggunakan *finite automata* dan *ekspresi reguler*. Misalnya, pengenalan (identifikasi), kata kunci (keyword), dan operator dalam bahasa pemrograman dapat didefinisikan menggunakan ekspresi reguler, dan FA digunakan untuk mengenali token-token ini dalam aliran masukan (Aho et al., 2007).
- **Analisis Sintaksis (Parsing):** Setelah token dihasilkan, tahap selanjutnya adalah analisis sintaksis, atau *parsing*. Pada tahap ini, struktur tata bahasa dari kode sumber diperiksa untuk memastikan bahwa ia sesuai dengan aturan tata bahasa bahasa pemrograman. *Bahasa bebas konteks* dan *pushdown automata* adalah alat utama yang digunakan dalam parsing. Pohon parse (parse tree) atau pohon sintaksis abstrak (abstract syntax tree) dibangun untuk merepresentasikan struktur hierarkis dari program, yang kemudian digunakan untuk tahap-tahap kompilasi selanjutnya.

1.1.2.2 Desain Bahasa Pemrograman

Prinsip-prinsip TBA juga memandu desain bahasa pemrograman itu sendiri. Pemilihan struktur sintaksis dan semantik suatu bahasa sangat dipengaruhi oleh teori bahasa formal. Desainer bahasa

harus memastikan bahwa bahasa yang mereka buat dapat diimplementasikan secara efisien oleh kompilator atau interpreter, yang berarti tata bahasanya harus dapat dianalisis oleh automata yang sesuai. Misalnya, sebagian besar bahasa pemrograman modern dirancang agar tata bahasanya bebas konteks, memungkinkan parsing yang efisien.

1.1.2.3 Pemrosesan Bahasa Alami (NLP)

Dalam bidang *Pemrosesan Bahasa Alami* (NLP), TBA menyediakan kerangka kerja untuk menganalisis dan memahami struktur bahasa manusia. Meskipun bahasa alami jauh lebih kompleks dan ambigu daripada bahasa formal, konsep-konsep dari TBA, seperti tata bahasa formal dan parsing, telah diadaptasi untuk tugas-tugas NLP. Misalnya, tata bahasa bebas konteks sering digunakan untuk memodelkan struktur sintaksis kalimat, dan algoritma parsing digunakan untuk mengurai kalimat menjadi komponen-komponennya (Jurafsky & Martin, 2009). Kemajuan terbaru dalam NLP, meskipun banyak didorong oleh pembelajaran mesin, masih memanfaatkan fondasi teoretis yang disediakan oleh TBA untuk pemahaman struktural.

1.1.2.4 Verifikasi Perangkat Lunak dan Perangkat Keras

Verifikasi formal adalah area lain di mana TBA memainkan peran krusial. Dalam pengembangan sistem kritis, seperti perangkat lunak penerbangan atau perangkat keras medis, sangat penting untuk memastikan bahwa sistem tersebut berperilaku sesuai spesifikasi. TBA menyediakan alat untuk memodelkan perilaku sistem sebagai automata dan kemudian menggunakan teknik verifikasi formal untuk membuktikan properti tertentu tentang sistem tersebut. Misalnya, *model checking* adalah teknik verifikasi yang menggunakan automata untuk memeriksa apakah suatu sistem memenuhi spesifikasi temporal yang diberikan (Clarke et al., 1999).

1.1.2.5 Keamanan Komputer

Dalam keamanan komputer, TBA digunakan untuk memodelkan dan menganalisis perilaku sistem yang rentan terhadap serangan. Misalnya, pola serangan atau *malware* dapat direpresentasikan sebagai bahasa formal, dan automata dapat digunakan untuk mendeteksi keberadaan pola-pola ini dalam aliran data atau kode. *Intrusion detection systems* (IDS) sering menggunakan teknik yang terinspirasi dari TBA untuk mengenali anomali atau pola serangan yang telah diketahui.

1.1.3 Batasan Komputasi dan Komputabilitas

Salah satu kontribusi paling mendalam dari TBA adalah pemahamannya tentang *batasan komputasi* dan *komputabilitas*. Konsep *Turing machine* (TM), yang diperkenalkan oleh Alan Turing, adalah model komputasi universal yang secara luas diyakini dapat mensimulasikan algoritma komputasi apa pun (Turing, 1936). Ini membentuk dasar dari *Tesis Church-Turing*, yang menyatakan bahwa setiap fungsi yang dapat dihitung secara efektif dapat dihitung oleh Turing machine.

Melalui studi tentang Turing machine, TBA telah mengidentifikasi kelas masalah yang *tidak dapat dipecahkan* (undecidable problems). Masalah-masalah ini, seperti *halting problem* (masalah penghentian), secara fundamental tidak dapat diselesaikan oleh algoritma komputasi apa pun, tidak peduli seberapa kuat mesinnya. Pemahaman tentang batasan ini sangat penting karena mencegah upaya yang sia-sia untuk mencari solusi algoritmik untuk masalah yang secara inheren tidak dapat dipecahkan. Ini juga menyoroti pentingnya merumuskan masalah dengan hati-hati dan memahami sifat komputasinya sebelum mencoba menyelesaikannya.

1.1.4 Komputasi Paralel dan Terdistribusi

Meskipun TBA secara tradisional berfokus pada model komputasi sekuensial, prinsip-prinsipnya juga relevan dalam konteks *komputasi paralel* dan *terdistribusi*. Konsep-konsep seperti

konkurensi, sinkronisasi, dan komunikasi antarproses dapat dimodelkan menggunakan ekstensi dari automata, seperti *petri nets* atau *process algebras*. Model-model ini memungkinkan analisis formal terhadap perilaku sistem paralel, membantu dalam mendeteksi *deadlock*, *livelock*, dan masalah konkurensi lainnya yang sulit diidentifikasi dalam sistem yang kompleks.

1.1.5 Pembelajaran Mesin dan Kecerdasan Buatan

Dalam beberapa tahun terakhir, ada peningkatan minat dalam menjembatani kesenjangan antara TBA dan *pembelajaran mesin* (machine learning) serta *kecerdasan buatan* (artificial intelligence). Meskipun pembelajaran mesin modern seringkali didominasi oleh pendekatan statistik dan jaringan saraf, fondasi formal dari TBA dapat memberikan wawasan tentang kemampuan dan batasan model-model ini. Misalnya, penelitian tentang *automata pembelajaran* (learning automata) mengeksplorasi bagaimana automata dapat belajar dari lingkungan mereka, sementara teori kompleksitas komputasi, yang berakar pada TBA, memberikan batasan teoretis pada efisiensi algoritma pembelajaran. Selain itu, dalam bidang *AI simbolik*, representasi pengetahuan dan penalaran seringkali memanfaatkan struktur formal yang mirip dengan bahasa formal.

1.1.6 Tantangan dan Arah Masa Depan

Meskipun TBA telah memberikan kontribusi yang luar biasa, bidang ini terus berkembang untuk mengatasi tantangan baru. Salah satu tantangan utama adalah menjembatani kesenjangan antara teori dan praktik, terutama dalam menghadapi kompleksitas sistem komputasi modern. Penelitian terus berlanjut dalam mengembangkan model automata yang lebih ekspresif dan efisien untuk memodelkan sistem yang kompleks, seperti sistem *cyber-physical* atau *quantum computing*.

Arah masa depan TBA juga mencakup eksplorasi *komputasi non-konvensional*, seperti *DNA computing* atau *neural computing*, di mana prinsip-prinsip komputasi mungkin berbeda secara

fundamental dari model Turing klasik. Memahami batasan dan kemampuan model-model baru ini akan memerlukan ekstensi dan modifikasi dari kerangka kerja TBA yang ada. Selain itu, integrasi yang lebih erat dengan bidang-bidang seperti pembelajaran mesin dan ilmu data akan terus menjadi area penelitian yang aktif, mencari cara untuk memanfaatkan kekuatan formal TBA untuk meningkatkan pemahaman dan pengembangan sistem cerdas.

Secara keseluruhan, ruang lingkup Teori Bahasa dan Automata sangat luas dan terus berkembang. Dari fondasi teoretis yang mendalam hingga aplikasi praktis yang tak terhitung jumlahnya, TBA tetap menjadi pilar penting dalam ilmu komputer, memberikan wawasan fundamental tentang sifat komputasi dan batasan-batasannya.

1.2. Peran Bahasa Formal dalam Ilmu Komputer

Bahasa formal merupakan tulang punggung ilmu komputer modern, menyediakan kerangka kerja yang presisi dan tidak ambigu untuk mendefinisikan, menganalisis, dan membangun sistem komputasi. Dalam esensinya, bahasa formal adalah sistem simbol dan aturan yang digunakan untuk merepresentasikan informasi dan proses secara matematis. Berbeda dengan bahasa alami yang kaya akan ambiguitas dan konteks, bahasa formal dirancang untuk kejelasan dan konsistensi, menjadikannya alat yang tak tergantikan dalam berbagai aspek komputasi.

1.2.1 Fondasi Teoretis untuk Pemrograman

Peran paling fundamental dari bahasa formal dalam ilmu komputer terletak pada penyediaan fondasi teoretis untuk bahasa pemrograman. Setiap bahasa pemrograman, mulai dari bahasa tingkat rendah seperti *assembly* hingga bahasa tingkat tinggi seperti Python atau Java, pada dasarnya adalah bahasa formal. Sintaksis dan semantik dari bahasa-bahasa ini didefinisikan secara ketat menggunakan konsep-konsep dari teori bahasa formal. Sintaksis, yang mengatur struktur dan bentuk ekspresi yang valid, sering kali dijelaskan menggunakan tata bahasa formal seperti

Backus-Naur Form (BNF) atau *Extended Backus-Naur Form* (EBNF). Semantik, yang mendefinisikan makna dari ekspresi yang valid, dapat dijelaskan melalui berbagai pendekatan formal, termasuk semantik operasional, denotasional, atau aksiomatik (Nielson & Nielson, 2019).

Sebagai contoh, pertimbangkan sebuah pernyataan sederhana dalam bahasa pemrograman: $x = y + z;$. Sintaksisnya akan menentukan bahwa sebuah variabel (seperti x), diikuti oleh operator penugasan ($=$), diikuti oleh ekspresi (seperti $y + z$), dan diakhiri dengan titik koma ($;$) adalah konstruksi yang valid. Semantiknya akan menjelaskan bahwa nilai dari ekspresi $y + z$ akan dihitung dan kemudian disimpan ke dalam lokasi memori yang diasosiasikan dengan variabel x . Tanpa definisi formal ini, kompiler atau interpreter tidak akan dapat memahami dan mengeksekusi kode program dengan benar.

1.2.2 Verifikasi dan Validasi Sistem

Salah satu tantangan terbesar dalam pengembangan perangkat lunak adalah memastikan kebenaran dan keandalan sistem. Bahasa formal memainkan peran krusial dalam verifikasi dan validasi sistem, terutama untuk sistem kritis di mana kegagalan dapat memiliki konsekuensi yang serius. Dengan menggunakan bahasa formal, spesifikasi sistem dapat ditulis secara presisi, menghilangkan ambiguitas yang sering muncul dalam spesifikasi bahasa alami. Spesifikasi formal ini kemudian dapat dianalisis menggunakan teknik-teknik seperti *model checking* atau *theorem proving* untuk secara matematis membuktikan bahwa sistem memenuhi properti yang diinginkan (Clarke et al., 2018).

Misalnya, dalam pengembangan sistem kontrol penerbangan, spesifikasi formal dapat digunakan untuk menyatakan bahwa pesawat tidak akan pernah berada di dua lokasi yang berbeda pada waktu yang sama atau setelah perintah pendaratan diberikan, roda pendaratan akan selalu terbuka dalam waktu X detik. Dengan menggunakan alat verifikasi formal, para insinyur dapat secara otomatis memeriksa apakah desain sistem yang diusulkan

memenuhi properti-properti keamanan dan keandalan ini sebelum implementasi fisik dilakukan. Pendekatan ini secara signifikan mengurangi risiko *bug* dan kesalahan desain yang mahal.

1.2.3 Desain Kompiler dan Interpreter

Desain kompiler dan interpreter adalah area lain di mana bahasa formal sangat diperlukan. Kompiler adalah program yang menerjemahkan kode sumber yang ditulis dalam satu bahasa pemrograman ke dalam kode objek atau kode mesin. Interpreter adalah program yang mengeksekusi kode sumber secara langsung. Kedua proses ini sangat bergantung pada teori bahasa formal.

Proses kompilasi biasanya melibatkan beberapa tahapan, di antaranya adalah analisis leksikal, analisis sintaksis, analisis semantik, dan pembangkitan kode. Analisis leksikal menggunakan konsep *finite automata* untuk mengidentifikasi token (kata kunci, pengenal, operator, dll.) dalam kode sumber. Analisis sintaksis menggunakan *context-free grammars* (CFG) dan *pushdown automata* untuk memeriksa struktur sintaksis kode dan membangun pohon parse. Analisis semantik, meskipun lebih kompleks, juga sering kali memanfaatkan aturan-aturan formal untuk memeriksa konsistensi tipe dan makna ekspresi. Tanpa kerangka kerja yang disediakan oleh teori bahasa formal, pembangunan kompiler yang efisien dan benar akan menjadi tugas yang jauh lebih sulit, jika bukan tidak mungkin (Aho et al., 2020).

1.2.4 Pemrosesan Bahasa Alami (NLP)

Meskipun bahasa formal dirancang untuk menghindari ambiguitas bahasa alami, prinsip-prinsip dan teknik dari teori bahasa formal telah menemukan aplikasi yang signifikan dalam bidang pemrosesan bahasa alami (NLP). Dalam NLP, tujuannya adalah untuk memungkinkan komputer memahami, menginterpretasi, dan menghasilkan bahasa manusia. Meskipun bahasa alami jauh lebih kompleks daripada bahasa formal, konsep-konsep seperti tata bahasa formal (misalnya, *context-free grammars* untuk analisis

sintaksis) dan *finite automata* (untuk pengenalan pola dalam teks) sering digunakan sebagai dasar.

Sebagai contoh, dalam *parsing* sintaksis kalimat bahasa alami, *context-free grammars* dapat digunakan untuk merepresentasikan struktur kalimat dan mengidentifikasi frasa nomina, frasa verba, dan komponen sintaksis lainnya. Meskipun tata bahasa ini seringkali perlu diperluas dengan aturan-aturan probabilistik atau statistik untuk menangani ambiguitas yang melekat pada bahasa alami, fondasi formalnya tetap berasal dari teori bahasa formal (Jurafsky & Martin, 2023).

1.2.5 Keamanan Komputer dan Kriptografi

Dalam keamanan komputer, bahasa formal digunakan untuk memodelkan dan menganalisis protokol keamanan, sistem otentikasi, dan kebijakan akses. Dengan merepresentasikan sistem keamanan sebagai model formal, para peneliti dapat menggunakan teknik verifikasi formal untuk mengidentifikasi kerentanan potensial, membuktikan properti keamanan, dan memastikan bahwa protokol berfungsi seperti yang diharapkan. Misalnya, protokol kriptografi dapat dimodelkan menggunakan bahasa formal untuk menganalisis kekuatan dan kelemahannya terhadap berbagai jenis serangan (Ryan et al., 2021).

Selain itu, dalam desain bahasa pemrograman yang aman, bahasa formal membantu dalam mendefinisikan semantik yang ketat untuk fitur-fitur keamanan, seperti kontrol akses dan manajemen memori. Hal ini memungkinkan pengembangan alat analisis statis yang dapat secara otomatis mendeteksi kerentanan keamanan dalam kode program sebelum dieksekusi.

1.2.6 Basis Data dan Bahasa Kueri

Bahasa formal juga merupakan inti dari sistem basis data dan bahasa kueri. Bahasa kueri seperti SQL (Structured Query Language) adalah contoh bahasa formal yang digunakan untuk berinteraksi dengan basis data relasional. Sintaksis dan semantik

SQL didefinisikan secara formal, memungkinkan pengguna untuk secara presisi mengambil, memanipulasi, dan mengelola data.

Teori di balik basis data relasional, termasuk aljabar relasional dan kalkulus relasional, adalah contoh lain dari aplikasi bahasa formal. Konsep-konsep ini menyediakan dasar matematis untuk mendefinisikan operasi pada data dan memastikan konsistensi serta integritas data. Tanpa kerangka kerja formal ini, basis data modern tidak akan dapat beroperasi dengan keandalan dan efisiensi yang kita kenal (Silberschatz et al., 2020).

1.2.7 Komputasi Terdistribusi dan Paralel

Dalam komputasi terdistribusi dan paralel, di mana banyak prosesor atau komputer bekerja sama untuk menyelesaikan tugas, bahasa formal digunakan untuk memodelkan perilaku sistem, mengidentifikasi kondisi balapan (race conditions), dan memastikan konsistensi data. Model formal seperti *Petri nets* atau *process calculi* (misalnya, *calculus of communicating systems* - CCS) digunakan untuk merepresentasikan interaksi antar proses dan menganalisis properti seperti *deadlock* dan *liveness*.

Dengan menggunakan bahasa formal, para pengembang dapat merancang dan memverifikasi algoritma terdistribusi yang kompleks, memastikan bahwa mereka beroperasi dengan benar bahkan dalam menghadapi kegagalan atau konkurensi yang tinggi. Ini sangat penting dalam sistem *cloud computing*, jaringan sensor, dan aplikasi terdistribusi lainnya di mana keandalan dan kinerja adalah kunci.

1.2.8 Kesimpulan

Secara keseluruhan, peran bahasa formal dalam ilmu komputer sangatlah luas dan mendalam. Dari fondasi teoretis bahasa pemrograman hingga verifikasi sistem kritis, desain kompiler, pemrosesan bahasa alami, keamanan komputer, basis data, hingga komputasi terdistribusi, bahasa formal menyediakan presisi, konsistensi, dan kerangka kerja analitis yang esensial. Kemampuannya untuk menghilangkan ambiguitas dan

memungkinkan analisis matematis menjadikan bahasa formal sebagai alat yang tak tergantikan bagi para ilmuwan komputer dan insinyur dalam merancang, membangun, dan menganalisis sistem komputasi yang kompleks dan andal. Pemahaman yang kuat tentang bahasa formal tidak hanya penting untuk memahami cara kerja sistem komputasi, tetapi juga untuk berinovasi dan mengembangkan teknologi komputasi di masa depan.

1.3. Sejarah dan Perkembangan Teori Automata

Teori automata, sebagai fondasi formal untuk komputasi modern, memiliki sejarah yang kaya dan perkembangan yang berkelanjutan, berakar pada upaya manusia untuk memahami dan mereplikasi proses berpikir dan perhitungan. Perjalanan ini dimulai jauh sebelum era komputer digital, dengan konsep-konsep abstrak yang secara bertahap diformalkan menjadi model-model matematis yang kuat. Memahami evolusi teori automata tidak hanya memberikan apresiasi terhadap kecerdasan para pionir di bidang ini, tetapi juga menyoroti bagaimana ide-ide fundamental ini terus membentuk lanskap teknologi informasi saat ini.

1.3.1 Akar Filosofis dan Matematis

Cikal bakal teori automata dapat ditelusuri kembali ke pemikiran filosofis tentang mekanisme dan logika. Filsuf seperti Gottfried Wilhelm Leibniz pada abad ke-17 membayangkan kalkulus rasional yang dapat menyelesaikan semua perselisihan melalui perhitungan mekanis, sebuah gagasan yang secara implisit mengandung konsep automata. Namun, formalisasi yang lebih konkret mulai muncul pada abad ke-19 dengan perkembangan logika simbolik. George Boole, dengan aljabar Boolean-nya, menyediakan kerangka kerja untuk merepresentasikan operasi logika secara matematis, yang kemudian menjadi dasar bagi desain sirkuit digital dan, secara tidak langsung, mesin komputasi.

Pada awal abad ke-20, David Hilbert mengajukan program formalisme untuk matematika, yang bertujuan untuk membangun semua matematika di atas sistem aksiomatik yang konsisten dan

lengkap. Program ini memicu pertanyaan mendalam tentang batasan komputasi dan provabilitas, yang pada akhirnya mengarah pada karya-karya fundamental dalam teori automata. Salah satu pertanyaan kunci yang diajukan adalah *Entscheidungsproblem* (masalah keputusan), yang menanyakan apakah ada algoritma umum yang dapat memutuskan kebenaran atau kepalsuan dari setiap pernyataan matematis.

1.3.2 Mesin Turing dan Fondasi Komputasi

Titik balik krusial dalam sejarah teori automata datang pada tahun 1936 dengan publikasi karya Alan Turing, *On Computable Numbers, with an Application to the Entscheidungsproblem*. Dalam karyanya, Turing memperkenalkan konsep *Mesin Turing*, sebuah model matematis abstrak dari sebuah perangkat komputasi. Mesin Turing, meskipun sederhana dalam strukturnya (terdiri dari pita tak terbatas, kepala baca/tulis, dan serangkaian keadaan), terbukti sangat kuat. Turing menunjukkan bahwa mesin ini mampu melakukan setiap komputasi yang dapat dilakukan oleh algoritma apa pun. Ini adalah pernyataan yang sangat mendalam, yang kemudian dikenal sebagai *Tesis Church-Turing*, yang menyatakan bahwa kelas fungsi yang dapat dihitung secara intuitif sama dengan kelas fungsi yang dapat dihitung oleh Mesin Turing.

Pentingnya Mesin Turing tidak dapat dilebih-lebihkan. Ini bukan hanya model teoretis; ia menyediakan cetak biru konseptual untuk komputer digital modern. Setiap komputer yang kita gunakan saat ini, pada dasarnya, adalah implementasi fisik dari Mesin Turing universal. Karya Turing tidak hanya menyelesaikan *Entscheheidungsproblem* dengan menunjukkan bahwa tidak ada algoritma umum untuk masalah tersebut, tetapi juga meletakkan dasar bagi seluruh bidang ilmu komputer, termasuk teori komputabilitas, kompleksitas komputasi, dan desain bahasa pemrograman.

1.3.3 Perkembangan Automata Finit dan Bahasa Formal

Seiring dengan perkembangan Mesin Turing, para peneliti juga mulai mengeksplorasi model komputasi yang lebih sederhana, yang dikenal sebagai *automata finit* (finite automata). Automata finit adalah model matematis dari mesin yang memiliki jumlah keadaan terbatas dan transisi antar keadaan yang ditentukan oleh input. Konsep ini pertama kali muncul dalam konteks pemodelan sistem saraf oleh Warren McCulloch dan Walter Pitts pada tahun 1943, yang mengusulkan model neuron buatan sebagai automata biner.

Pada tahun 1950-an, Stephen Kleene secara formal mendefinisikan *ekspresi reguler* dan menunjukkan ekuivalensinya dengan automata finit. Ekspresi reguler adalah notasi yang ringkas untuk mendeskripsikan pola dalam string, dan hingga saat ini, ekspresi reguler menjadi alat yang sangat penting dalam berbagai aplikasi, mulai dari pencarian teks hingga kompilasi bahasa pemrograman.

Noam Chomsky, seorang ahli bahasa, pada tahun 1950-an dan 1960-an, memberikan kontribusi monumental terhadap teori bahasa formal dengan memperkenalkan *hierarki Chomsky*. Hierarki ini mengklasifikasikan bahasa formal ke dalam empat tingkatan berdasarkan kompleksitas tata bahasanya dan jenis automata yang dapat mengenalinya:

- **Bahasa Tipe 0 (Bahasa Rekursif Enumerable):** Dikenali oleh Mesin Turing. Ini adalah kelas bahasa yang paling umum dan kuat.
- **Bahasa Tipe 1 (Bahasa Konteks-Sensitif):** Dikenali oleh automata linear-bounded. Tata bahasa ini memungkinkan aturan produksi yang bergantung pada konteks simbol.
- **Bahasa Tipe 2 (Bahasa Konteks-Bebas):** Dikenali oleh automata pushdown. Tata bahasa ini sangat penting dalam desain bahasa pemrograman dan parsing.
- **Bahasa Tipe 3 (Bahasa Reguler):** Dikenali oleh automata finit. Ini adalah kelas bahasa yang paling sederhana dan paling banyak digunakan dalam aplikasi praktis.

Hierarki Chomsky memberikan kerangka kerja yang sistematis untuk memahami hubungan antara bahasa formal dan model komputasi. Ini juga menjadi dasar bagi pengembangan kompilator dan parser, yang merupakan komponen inti dari setiap bahasa pemrograman.

1.3.4 Automata Pushdown dan Kompilasi

Pengembangan *automata pushdown* pada tahun 1960-an merupakan langkah penting lainnya dalam evolusi teori automata. Automata pushdown adalah automata finit yang diperluas dengan tumpukan (stack) memori. Kemampuan untuk menyimpan dan mengambil simbol dari tumpukan memungkinkan automata pushdown untuk mengenali bahasa konteks-bebas, yang jauh lebih kompleks daripada bahasa reguler.

Pentingnya automata pushdown terletak pada perannya dalam kompilasi bahasa pemrograman. Sebagian besar bahasa pemrograman modern, seperti C++, Java, dan Python, memiliki tata bahasa yang bersifat konteks-bebas. Kompilator menggunakan automata pushdown (atau varian yang lebih canggih) untuk menganalisis struktur sintaksis kode sumber dan mengubahnya menjadi kode mesin yang dapat dieksekusi. Tanpa pemahaman tentang automata pushdown dan bahasa konteks-bebas, pengembangan kompilator yang efisien akan menjadi tugas yang jauh lebih sulit.

1.3.5 Perkembangan Modern dan Aplikasi Kontemporer

Sejak fondasi-fondasi ini diletakkan, teori automata terus berkembang dan menemukan aplikasi baru di berbagai bidang. Pada tahun-tahun terakhir, penelitian telah bergeser ke arah model automata yang lebih kompleks dan spesifik, seperti *automata berwaktu* (timed automata) untuk sistem real-time, *automata hibrida* (hybrid automata) untuk sistem yang menggabungkan perilaku diskrit dan kontinu, dan *automata kuantum* (quantum automata) untuk komputasi kuantum yang sedang berkembang.

Aplikasi teori automata saat ini sangat luas dan beragam:

- **Desain Kompilator dan Bahasa Pemrograman:** Seperti yang telah disebutkan, automata finit dan pushdown adalah tulang punggung dari setiap kompilator dan parser.
- **Verifikasi Perangkat Lunak dan Perangkat Keras:** Automata digunakan untuk memodelkan perilaku sistem dan memverifikasi bahwa sistem tersebut memenuhi spesifikasi yang diinginkan. Misalnya, model checking, sebuah teknik verifikasi formal, sangat bergantung pada teori automata untuk mengeksplorasi semua kemungkinan keadaan sistem.
- **Pemrosesan Bahasa Alami (NLP):** Meskipun NLP modern banyak menggunakan teknik pembelajaran mesin, konsep-konsep dari teori bahasa formal masih relevan dalam analisis sintaksis dan semantik.
- **Bioinformatika:** Automata digunakan untuk memodelkan struktur DNA dan RNA, serta untuk menganalisis urutan genetik.
- **Keamanan Siber:** Automata dapat digunakan untuk mendeteksi pola serangan dalam lalu lintas jaringan atau untuk menganalisis perilaku malware.
- **Robotika dan Sistem Otonom:** Automata digunakan untuk merancang pengontrol untuk robot dan sistem otonom, memungkinkan mereka untuk berinteraksi dengan lingkungan dan membuat keputusan.
- **Teori Permainan:** Automata dapat digunakan untuk memodelkan strategi pemain dalam permainan dan menganalisis hasil yang mungkin.

Sebagai contoh konkret, pertimbangkan penggunaan automata finit dalam *regular expressions* (regex). Regex adalah alat yang sangat ampuh untuk pencarian pola dalam teks. Ketika Anda menggunakan regex untuk mencari alamat email dalam dokumen, di balik layar, mesin regex mengonversi pola tersebut menjadi automata finit dan kemudian menggunakannya untuk memindai teks secara efisien. Ini adalah contoh bagaimana konsep abstrak

dari teori automata memiliki dampak langsung pada alat yang kita gunakan setiap hari.

Contoh lain adalah dalam *model checking* untuk verifikasi perangkat lunak. Misalkan kita memiliki sistem perangkat lunak kritis, seperti sistem kontrol penerbangan. Kita dapat memodelkan perilaku sistem ini sebagai automata, dan kemudian menggunakan algoritma model checking (yang didasarkan pada teori automata) untuk secara otomatis memeriksa apakah sistem tersebut dapat mencapai keadaan yang tidak aman. Ini membantu memastikan keandalan dan keamanan sistem yang kompleks.

Perkembangan teori automata terus berlanjut, didorong oleh tantangan baru dalam komputasi dan kebutuhan akan model yang lebih canggih untuk sistem yang semakin kompleks. Dari akar filosofis hingga aplikasi kontemporer, teori automata tetap menjadi bidang yang dinamis dan fundamental dalam ilmu komputer.

1.4. Hubungan Teori Bahasa dengan Algoritma dan Komputasi

Teori Bahasa Formal (TLF) dan Teori Automata (TA) adalah fondasi teoretis yang tak terpisahkan dari ilmu komputer modern. Hubungan antara TLF, TA, algoritma, dan komputasi sangat fundamental, membentuk kerangka kerja untuk memahami batasan dan kemampuan komputasi. Algoritma, sebagai serangkaian instruksi yang terdefinisi dengan baik untuk menyelesaikan masalah, secara inheren terkait dengan konsep bahasa formal. Setiap algoritma dapat dipandang sebagai representasi dari suatu bahasa formal, di mana input dan output algoritma adalah kata-kata dalam bahasa tersebut, dan proses komputasi adalah aturan tata bahasa yang mengubah satu kata menjadi kata lain.

1.4.1 Bahasa Formal sebagai Model Algoritma

Dalam esensinya, bahasa formal menyediakan kerangka kerja yang presisi untuk mendefinisikan dan menganalisis algoritma. Sebuah

algoritma dapat dianggap sebagai sebuah fungsi yang memetakan input dari suatu bahasa formal ke output dalam bahasa formal lainnya. Misalnya, dalam kompilasi, kode sumber (sebuah string dalam bahasa pemrograman formal) diubah menjadi kode objek (sebuah string dalam bahasa mesin formal). Proses ini melibatkan serangkaian langkah algoritmik yang secara ketat didefinisikan oleh aturan sintaksis dan semantik bahasa pemrograman.

Konsep *mesin Turing*, yang merupakan model komputasi paling kuat, secara langsung terkait dengan bahasa formal. Mesin Turing dapat mengenali *bahasa rekursif* (juga dikenal sebagai *bahasa yang dapat dihitung secara rekursif*), yang merupakan kelas bahasa formal yang paling umum dan mencakup sebagian besar bahasa pemrograman praktis. Kemampuan mesin Turing untuk mensimulasikan algoritma apa pun yang dapat diimplementasikan pada komputer modern menunjukkan bahwa bahasa formal yang dapat dikenali oleh mesin Turing adalah representasi yang kuat untuk masalah-masalah yang dapat dipecahkan secara algoritmik.

1.4.2 Klasifikasi Bahasa Formal dan Kompleksitas Algoritma

Hierarki Chomsky, yang mengklasifikasikan bahasa formal berdasarkan kompleksitas tata bahasanya, memiliki implikasi langsung terhadap kompleksitas algoritma yang diperlukan untuk memproses bahasa-bahasa tersebut.

- **Bahasa Regular:** Bahasa regular adalah yang paling sederhana dalam Hierarki Chomsky dan dapat dikenali oleh *finite automata* (FA). Algoritma untuk memproses bahasa regular, seperti pencarian pola dalam teks menggunakan ekspresi reguler, sangat efisien dan memiliki kompleksitas waktu linier. Contohnya adalah validasi format email atau nomor telepon.
- **Bahasa Bebas Konteks:** Bahasa bebas konteks (CFG) lebih kompleks daripada bahasa regular dan dapat dikenali oleh *pushdown automata* (PDA). CFG digunakan secara luas dalam mendefinisikan sintaksis bahasa pemrograman. Algoritma untuk mem-parsing bahasa bebas konteks, seperti algoritma CYK atau Earley, memiliki kompleksitas polinomial. Meskipun

lebih kompleks daripada bahasa regular, bahasa bebas konteks masih dapat diproses secara efisien untuk sebagian besar aplikasi praktis.

- **Bahasa Sensitif Konteks:** Bahasa sensitif konteks (CSG) lebih kompleks lagi dan membutuhkan *linear-bounded automata* (LBA) untuk mengenalannya. CSG jarang digunakan secara langsung dalam desain bahasa pemrograman karena kompleksitas parsingnya yang tinggi. Namun, konsep-konsep dari CSG relevan dalam analisis semantik bahasa alami dan beberapa aspek kompilasi yang lebih canggih.
- **Bahasa Rekursif (Recursively Enumerable Languages):** Ini adalah kelas bahasa yang paling umum dan dapat dikenali oleh mesin Turing. Bahasa rekursif mencakup semua bahasa yang dapat diproses oleh algoritma apa pun. Masalah *halting problem* adalah contoh klasik dari masalah yang tidak dapat dipecahkan oleh mesin Turing, menunjukkan batasan fundamental komputasi dan, secara tidak langsung, batasan bahasa rekursif.

Hubungan antara Hierarki Chomsky dan kompleksitas algoritma sangat penting dalam desain bahasa pemrograman dan kompilator. Pemilihan kelas bahasa yang tepat untuk suatu tugas komputasi akan secara langsung memengaruhi efisiensi dan kelayakan implementasi algoritmik.

1.4.3 Teori Komputabilitas dan Batasan Algoritma

Teori bahasa formal, khususnya melalui model mesin Turing, memberikan dasar untuk teori komputabilitas, yang mempelajari apa yang dapat dan tidak dapat dihitung oleh algoritma. Konsep *undecidability* (ketidakputusan) adalah salah satu hasil paling mendalam dari teori komputabilitas. Masalah yang tidak dapat diputuskan adalah masalah yang tidak ada algoritma yang dapat menyelesaikannya untuk semua kemungkinan input.

Contoh paling terkenal adalah *halting problem*, yang menyatakan bahwa tidak ada algoritma umum yang dapat menentukan apakah program arbitrer akan berhenti atau berjalan selamanya untuk

input tertentu. Implikasi dari *halting problem* sangat luas, menunjukkan bahwa ada batasan fundamental pada apa yang dapat dicapai oleh komputasi algoritmik.

Studi tentang masalah yang tidak dapat diputuskan ini sangat relevan dalam pengembangan perangkat lunak dan desain sistem. Memahami batasan komputasi membantu para insinyur dan ilmuwan komputer untuk menghindari upaya yang sia-sia dalam mencari solusi algoritmik untuk masalah yang secara inheren tidak dapat dipecahkan. Sebaliknya, mereka dapat fokus pada menemukan solusi heuristik atau pendekatan yang dapat diterima untuk masalah-masalah tersebut.

1.4.4 Aplikasi dalam Desain Kompilator dan Interpreter

Salah satu aplikasi paling nyata dari teori bahasa formal dan automata dalam komputasi adalah desain kompilator dan interpreter. Kompilator adalah program yang menerjemahkan kode sumber yang ditulis dalam bahasa pemrograman tingkat tinggi ke dalam kode mesin atau kode perantara yang dapat dieksekusi oleh komputer. Proses ini secara fundamental didasarkan pada prinsip-prinsip teori bahasa formal.

- **Analisis Leksikal:** Tahap pertama dalam kompilasi adalah analisis leksikal, di mana kode sumber dipecah menjadi unit-unit dasar yang disebut *token*. Proses ini sering kali diimplementasikan menggunakan *finite automata* atau ekspresi reguler. Misalnya, pengenalan (variabel), kata kunci, operator, dan literal semuanya diidentifikasi sebagai token berdasarkan pola-pola reguler.
- **Analisis Sintaksis:** Setelah token diidentifikasi, tahap selanjutnya adalah analisis sintaksis (parsing), di mana struktur gramatikal kode sumber diperiksa. Tahap ini biasanya diimplementasikan menggunakan *pushdown automata* atau algoritma parsing yang didasarkan pada tata bahasa bebas konteks. Pohon parse (parse tree) atau pohon sintaksis abstrak (abstract syntax tree) dibangun untuk merepresentasikan struktur hierarkis kode.

- **Analisis Semantik:** Tahap analisis semantik memeriksa makna kode, memastikan bahwa program tersebut logis dan konsisten. Meskipun tidak secara langsung didasarkan pada automata formal, prinsip-prinsip teori bahasa formal memberikan kerangka kerja untuk mendefinisikan dan memverifikasi aturan semantik.
- **Generasi Kode:** Akhirnya, kompilator menghasilkan kode objek atau kode perantara. Meskipun tahap ini lebih berorientasi pada arsitektur mesin, pemahaman tentang bahasa formal membantu dalam merancang representasi perantara yang efisien dan dalam menghasilkan kode yang benar.

Penggunaan teori bahasa formal dalam kompilator tidak hanya memastikan kebenaran sintaksis dan semantik program, tetapi juga memungkinkan pengembangan alat bantu yang kuat seperti generator parser (misalnya, Yacc/Bison, ANTLR) yang secara otomatis menghasilkan parser dari spesifikasi tata bahasa formal. Ini secara signifikan mempercepat proses pengembangan kompilator dan mengurangi kemungkinan kesalahan.

1.4.5 Peran dalam Verifikasi Formal dan Rekayasa Perangkat Lunak

Selain kompilator, teori bahasa formal juga memainkan peran penting dalam verifikasi formal dan rekayasa perangkat lunak. Verifikasi formal adalah proses pembuktian kebenaran atau properti tertentu dari suatu sistem perangkat lunak atau perangkat keras menggunakan metode matematika yang ketat.

- **Model Checking:** Salah satu teknik verifikasi formal yang populer adalah *model checking*, di mana sistem dimodelkan sebagai *finite state machine* (FSM) dan properti yang diinginkan dinyatakan dalam logika temporal (sebuah jenis bahasa formal). Algoritma model checking kemudian secara sistematis memeriksa semua kemungkinan keadaan sistem untuk memverifikasi apakah properti tersebut berlaku. Aplikasi

model checking sangat luas, mulai dari verifikasi protokol komunikasi hingga sistem kontrol kritis.

- **Analisis Program Statis:** Teori bahasa formal juga mendukung analisis program statis, di mana properti program dianalisis tanpa benar-benar menjalankannya. Teknik seperti analisis aliran data dan analisis kontrol aliran sering kali menggunakan konsep-konsep dari teori automata untuk membangun model perilaku program dan mengidentifikasi potensi bug atau kerentanan.
- **Spesifikasi Formal:** Bahasa formal digunakan untuk menspesifikasikan persyaratan sistem secara presisi dan tidak ambigu. Ini membantu dalam mengurangi ambiguitas dalam spesifikasi dan memastikan bahwa implementasi memenuhi persyaratan yang dimaksud.

Dalam rekayasa perangkat lunak, pemahaman tentang teori bahasa formal memungkinkan pengembang untuk merancang sistem yang lebih robust, aman, dan dapat diandalkan. Dengan menggunakan alat dan teknik yang didasarkan pada teori ini, mereka dapat mengidentifikasi dan memperbaiki kesalahan pada tahap awal pengembangan, yang secara signifikan mengurangi biaya dan risiko proyek.

1.4.6 Bahasa Formal dalam Pemrosesan Bahasa Alami dan Kecerdasan Buatan

Meskipun bahasa formal secara inheren berbeda dari bahasa alami, prinsip-prinsip teori bahasa formal telah memberikan kontribusi signifikan pada bidang pemrosesan bahasa alami (NLP) dan kecerdasan buatan (AI).

- **Parsing Bahasa Alami:** Upaya awal dalam NLP banyak menggunakan tata bahasa bebas konteks untuk mem-parsing kalimat bahasa alami, meskipun dengan keterbatasan karena kompleksitas dan ambiguitas bahasa alami. Meskipun model statistik dan pembelajaran mesin kini mendominasi NLP, konsep-konsep dasar dari parsing formal masih relevan dalam memahami struktur sintaksis kalimat.

- **Representasi Pengetahuan:** Dalam AI, bahasa formal digunakan untuk merepresentasikan pengetahuan secara terstruktur dan logis. Logika predikat, misalnya, adalah bahasa formal yang kuat untuk merepresentasikan fakta dan aturan, memungkinkan sistem AI untuk melakukan inferensi dan penalaran.
- **Desain Bahasa Pemrograman AI:** Bahasa pemrograman seperti Lisp dan Prolog, yang secara historis penting dalam AI, memiliki akar yang kuat dalam teori bahasa formal dan logika. Mereka menyediakan mekanisme untuk memanipulasi simbol dan struktur data secara deklaratif, yang sangat cocok untuk tugas-tugas AI.

Meskipun tantangan dalam memodelkan kompleksitas bahasa alami dengan bahasa formal masih besar, fondasi yang diletakkan oleh teori bahasa formal terus menginformasikan penelitian dan pengembangan di bidang NLP dan AI, terutama dalam aspek-aspek yang berkaitan dengan struktur, sintaksis, dan logika.

Secara keseluruhan, hubungan antara teori bahasa formal, algoritma, dan komputasi adalah simbiotik dan mendalam. Teori bahasa formal menyediakan kerangka kerja matematis yang presisi untuk mendefinisikan, menganalisis, dan memahami algoritma, sementara algoritma adalah implementasi praktis dari konsep-konsep yang berasal dari teori bahasa formal. Pemahaman yang kuat tentang hubungan ini sangat penting bagi setiap ilmuwan komputer atau insinyur perangkat lunak untuk merancang, menganalisis, dan membangun sistem komputasi yang efisien, benar, dan andal.

1.5. Peta Konsep dan Struktur Pembahasan Buku

Buku ini dirancang untuk memberikan pemahaman yang komprehensif mengenai Teori Bahasa dan Automata, sebuah fondasi krusial dalam ilmu komputer modern. Struktur pembahasan disusun secara sistematis, dimulai dari konsep-konsep dasar hingga topik-topik yang lebih kompleks, dengan

tujuan agar pembaca dapat membangun pemahaman yang kokoh dan terstruktur. Peta konsep yang disajikan di sini akan menjadi panduan bagi pembaca untuk menelusuri alur materi, memahami keterkaitan antar bab, dan mengidentifikasi tujuan pembelajaran di setiap bagian.

1.5.1 Pendekatan Pedagogis dan Alur Pembelajaran

Pendekatan pedagogis dalam buku ini berpusat pada *pembelajaran bertahap* (scaffolding), di mana setiap bab dibangun di atas pengetahuan yang telah diperoleh sebelumnya. Kami percaya bahwa pemahaman yang mendalam tentang Teori Bahasa dan Automata tidak hanya memerlukan penguasaan definisi dan teorema, tetapi juga kemampuan untuk menerapkan konsep-konsep tersebut dalam berbagai skenario komputasi. Oleh karena itu, setiap bab akan dilengkapi dengan contoh-contoh konkret, studi kasus, dan latihan yang dirancang untuk memperkuat pemahaman teoritis dan mengembangkan keterampilan analitis pembaca.

Alur pembelajaran dimulai dengan pengenalan konsep-konsep fundamental seperti alfabet, string, dan bahasa formal, yang merupakan blok bangunan dasar dari seluruh teori. Dari sana, kita akan bergerak menuju pembahasan automata hingga mesin Turing, yang merepresentasikan puncak dari model komputasi. Setiap model automata akan dijelaskan secara rinci, termasuk definisi formal, cara kerja, kemampuan komputasi, dan keterbatasannya. Penekanan juga akan diberikan pada hubungan antara kelas-kelas bahasa formal dan model-model automata yang dapat mengenalinya, sebuah konsep sentral dalam hierarki Chomsky.

1.5.2 Struktur Bab dan Keterkaitan Materi

Buku ini terbagi menjadi beberapa bab utama, masing-masing membahas aspek spesifik dari Teori Bahasa dan Automata. Berikut adalah gambaran umum struktur bab dan bagaimana setiap bab saling terkait:

- **Bab 1: Pendahuluan Teori Bahasa dan Automata.** Bab ini berfungsi sebagai pengantar, memperkenalkan ruang lingkup Teori Bahasa dan Automata, perannya dalam ilmu komputer, sejarah perkembangannya, serta hubungannya dengan algoritma dan komputasi. Bab ini juga akan memaparkan peta konsep dan struktur pembahasan buku secara keseluruhan, memberikan gambaran awal kepada pembaca tentang perjalanan intelektual yang akan ditempuh.
- **Bab 2: Konsep Dasar Bahasa Formal.** Bab ini akan meletakkan dasar-dasar yang diperlukan untuk memahami materi selanjutnya. Pembahasan meliputi definisi alfabet, string, operasi pada string, dan konsep bahasa formal. Kita akan mengeksplorasi berbagai cara untuk mendefinisikan bahasa formal, termasuk melalui enumerasi, deskripsi properti, dan ekspresi reguler. Pemahaman yang kuat tentang konsep-konsep ini sangat penting karena mereka menjadi fondasi bagi semua model automata dan tata bahasa yang akan dibahas di bab-bab berikutnya.
- **Bab 3: Automata Hingga (Finite Automata).** Bab ini memperkenalkan model komputasi paling sederhana, yaitu automata hingga. Kita akan membahas dua jenis utama: Automata Hingga Deterministik (DFA) dan Automata Hingga Non-Deterministik (NFA). Penjelasan akan mencakup definisi formal, representasi grafis, dan cara kerja masing-masing. Hubungan antara DFA dan NFA, serta ekuivalensinya dalam mengenali bahasa reguler, akan menjadi fokus utama. Studi kasus akan melibatkan perancangan DFA/NFA untuk mengenali pola-pola tertentu dalam string, seperti validasi input sederhana atau pencarian teks.
- **Bab 4: Bahasa Reguler dan Ekspresi Reguler.** Melanjutkan dari Bab 3, bab ini akan membahas secara mendalam kelas bahasa yang dapat dikenali oleh automata hingga, yaitu bahasa reguler. Kita akan mengeksplorasi ekspresi reguler sebagai notasi yang ringkas untuk mendefinisikan bahasa reguler, serta hubungan timbal balik antara ekspresi reguler, NFA, dan DFA. Algoritma konversi antara ketiga representasi ini akan

dijelaskan secara detail. Teorema penting seperti *Pumping Lemma for Regular Languages* akan diperkenalkan untuk membuktikan bahwa suatu bahasa *bukan* bahasa reguler. Aplikasi praktis dari bahasa reguler, seperti dalam *lexical analysis* pada kompilator dan pencarian pola teks (misalnya, menggunakan *grep*), akan disajikan.

- **Bab 5: Tata Bahasa Bebas Konteks (Context-Free Grammars) dan Bahasa Bebas Konteks.** Bab ini memperkenalkan model tata bahasa yang lebih kuat daripada ekspresi reguler, yaitu tata bahasa bebas konteks (CFG). CFG digunakan untuk mendefinisikan bahasa bebas konteks (CFL), yang memiliki kemampuan deskriptif lebih tinggi. Kita akan membahas definisi formal CFG, derivasi, pohon parse, dan ambiguitas. Normalisasi CFG, seperti bentuk Chomsky Normal Form (CNF) dan Greibach Normal Form (GNF), juga akan dibahas. CFL sangat penting dalam sintaksis bahasa pemrograman dan parsing. Contoh-contoh akan mencakup definisi sintaksis ekspresi aritmatika dan struktur program sederhana.
- **Bab 6: Automata Dorong-Bawah (Pushdown Automata).** Bab ini memperkenalkan model automata yang setara dengan tata bahasa bebas konteks, yaitu automata dorong-bawah (PDA). PDA adalah automata hingga yang diperluas dengan sebuah *stack*, memberikannya kemampuan untuk mengingat informasi yang lebih kompleks. Kita akan membahas definisi formal PDA, cara kerja, dan hubungannya dengan CFG. Konversi antara CFG dan PDA akan dijelaskan. Aplikasi PDA dalam parsing bahasa pemrograman akan menjadi fokus utama, menunjukkan bagaimana PDA dapat digunakan untuk memvalidasi struktur sintaksis.
- **Bab 7: Mesin Turing.** Bab ini adalah puncak dari pembahasan model komputasi, memperkenalkan Mesin Turing (TM), yang secara universal diakui sebagai model komputasi paling kuat. TM dapat mensimulasikan algoritma apa pun yang dapat dihitung. Kita akan membahas definisi formal TM, cara kerja, dan konsep *Church-Turing Thesis* yang menyatakan bahwa TM

dapat menghitung semua fungsi yang dapat dihitung secara efektif. Berbagai varian TM, seperti TM multi-pita dan TM non-deterministik, akan dibahas, serta ekuivalensinya dengan TM standar. Konsep *halting problem* dan *undecidability* akan diperkenalkan, menunjukkan batasan fundamental dari komputasi.

- **Bab 8: Komputabilitas dan Kompleksitas.** Bab ini akan membahas lebih lanjut tentang batasan komputasi dan efisiensi algoritma. Konsep *decidability* dan *undecidability* akan dieksplorasi secara mendalam, dengan contoh-contoh masalah yang tidak dapat dipecahkan oleh algoritma. Selain itu, kita akan memperkenalkan teori kompleksitas komputasi, yang mengklasifikasikan masalah berdasarkan sumber daya (waktu dan ruang) yang dibutuhkan untuk menyelesaikannya. Kelas-kelas kompleksitas seperti P, NP, NP-lengkap, dan NP-sulit akan dibahas, memberikan pemahaman tentang tantangan dalam merancang algoritma yang efisien untuk masalah-masalah yang sulit.

1.5.3 Studi Kasus dan Contoh Konkret

Untuk memperkaya pemahaman, setiap bab akan menyertakan studi kasus dan contoh konkret yang relevan dengan aplikasi dunia nyata. Misalnya, dalam pembahasan automata hingga, kita akan melihat bagaimana mereka digunakan dalam perancangan sirkuit digital sederhana atau dalam implementasi mesin pencari teks. Ketika membahas tata bahasa bebas konteks, kita akan menganalisis bagaimana CFG digunakan untuk mendefinisikan sintaksis bahasa pemrograman seperti Python atau Java, dan bagaimana parser menggunakan CFG untuk memvalidasi dan menginterpretasikan kode sumber.

Contoh lain termasuk penggunaan ekspresi reguler dalam validasi format email atau nomor telepon, serta bagaimana konsep Mesin Turing menjadi dasar bagi arsitektur komputer modern dan pengembangan bahasa pemrograman tingkat tinggi. Studi kasus ini tidak hanya akan mengilustrasikan teori, tetapi juga akan

menunjukkan relevansi praktis dari Teori Bahasa dan Automata dalam berbagai bidang ilmu komputer, mulai dari perancangan kompilator, sistem operasi, hingga kecerdasan buatan.

1.5.4 Tujuan Pembelajaran

Setelah menyelesaikan buku ini, pembaca diharapkan memiliki kemampuan untuk:

- Memahami konsep dasar bahasa formal, alfabet, string, dan operasi-operasinya.
- Menganalisis dan merancang Automata Hingga (DFA dan NFA) untuk mengenali bahasa reguler.
- Menggunakan ekspresi reguler untuk mendefinisikan dan memanipulasi bahasa reguler.
- Memahami dan menerapkan Tata Bahasa Bebas Konteks (CFG) untuk mendefinisikan bahasa bebas konteks.
- Menganalisis dan merancang Automata Dorong-Bawah (PDA) untuk mengenali bahasa bebas konteks.
- Memahami prinsip kerja dan kemampuan komputasi Mesin Turing.
- Menganalisis masalah komputasi dalam konteks *decidability* dan *undecidability*.
- Memahami konsep dasar teori kompleksitas komputasi dan klasifikasi masalah (P, NP, NP-lengkap).
- Menerapkan konsep Teori Bahasa dan Automata untuk memecahkan masalah praktis dalam ilmu komputer.

Dengan struktur yang terorganisir dengan baik, pendekatan pedagogis yang berpusat pada pembaca, serta penekanan pada contoh dan aplikasi, buku ini bertujuan untuk menjadi sumber daya yang tak ternilai bagi mahasiswa, peneliti, dan praktisi yang ingin menguasai fondasi formal komputasi modern.

GLOSARIUM

- **Alfabet:** Himpunan berhingga dan tidak kosong dari simbol-simbol dasar yang digunakan untuk membangun entitas yang lebih kompleks, di mana setiap simbol adalah entitas atomik yang tidak dapat dipecah lagi;
- **Alfabet Input:** Himpunan terbatas dari semua simbol yang dapat menjadi bagian dari *string input* yang akan diproses oleh Finite Automata;
- **Alfabet Pita:** Himpunan simbol-simbol yang dapat ditulis pada pita Mesin Turing, mencakup alfabet input dan simbol-simbol kerja tambahan, termasuk simbol kosong (blank);
- **Ambiguitas:** Kondisi dalam Tata Bahasa Bebas Konteks (CFG) di mana setidaknya satu string dalam bahasanya memiliki dua atau lebih pohon derivasi yang berbeda, menyebabkan interpretasi sintaksis yang tidak tunggal;
- **Ambiguitas Grammar:** Kondisi di mana sebuah grammar formal memungkinkan setidaknya satu string dalam bahasanya untuk memiliki dua atau lebih pohon parse yang berbeda, yang dapat menyebabkan interpretasi yang beragam dari kode atau kalimat yang sama;
- **Analisis Leksikal:** Tahap pertama dalam pemrosesan bahasa pemrograman di mana kode sumber dipecah menjadi serangkaian unit dasar yang disebut token, mengabaikan spasi kosong dan komentar;
- **Analisis Sintaksis:** Tahap dalam pemrosesan bahasa pemrograman setelah analisis leksikal, di mana urutan token diperiksa kesesuaiannya dengan aturan tata bahasa bahasa pemrograman untuk membangun representasi hirarkis kode sumber;

- **Automata:** Model matematis dari mesin komputasi abstrak yang dirancang untuk memproses bahasa formal, berfungsi sebagai pengenal yang menentukan apakah suatu *string* masukan termasuk dalam bahasa tertentu;
- **Automata Hingga (Finite Automata):** Model matematika komputasi yang memiliki sejumlah keadaan terbatas dan digunakan untuk mengenali bahasa reguler;
- **Automata Hingga Berhingga (Finite Automata - FA):** Model komputasi abstrak yang digunakan secara luas dalam analisis leksikal untuk mengenali pola-pola token yang didefinisikan oleh ekspresi reguler;
- **Bahasa:** Himpunan string yang merupakan subset dari semua string yang dapat dibentuk dari suatu alfabet (Σ^*), di mana setiap string dalam bahasa tersebut harus terdiri dari simbol-simbol yang berasal dari alfabet Σ ;
- **Bahasa Bebas Konteks (Context-Free Language):** Kategori bahasa formal yang dihasilkan oleh tata bahasa bebas konteks (CFG), di mana aturan produksi hanya memiliki satu simbol non-terminal di sisi kiri, memungkinkan pemodelan struktur sintaksis bahasa pemrograman dan sebagian besar aspek sintaksis bahasa alami;
- **Bahasa Bebas Konteks (Context-Free Languages - CFL):** Kelas bahasa formal yang lebih luas dari bahasa reguler, yang dapat dikenali oleh Pushdown Automata dan cukup kuat untuk mendefinisikan sintaksis sebagian besar bahasa pemrograman;
- **Bahasa Bebas Konteks (Context-Free Languages - CFLs):** Kelas bahasa formal yang lebih luas dari bahasa reguler, yang dapat dikenali oleh Pushdown Automata dan dijelaskan oleh Context-Free Grammar;
- **Bahasa Formal:** Himpunan *string* yang dibentuk dari alfabet tertentu, mengikuti serangkaian aturan tata bahasa yang presisi dan didefinisikan secara matematis, berbeda dengan bahasa alami yang ambigu;
- **Bahasa Reguler:** Himpunan string yang dapat direpresentasikan oleh ekspresi reguler atau diterima oleh suatu automata hingga;

- **Bahasa Regular (Regular Language):** Kategori bahasa formal paling sederhana dalam Hirarki Chomsky, dihasilkan oleh tata bahasa reguler (RG) dengan aturan produksi yang sangat terbatas, relevan untuk pencarian pola teks dan analisis leksikal dalam kompilator;
- **Bahasa Rekursif (Decidable Languages):** Sebuah himpunan string yang dapat diputuskan keanggotaannya oleh Mesin Turing yang selalu berhenti untuk setiap input, baik menerima maupun menolak string tersebut;
- **Bahasa Rekursif Enumerable (Recognizable Languages):** Sebuah himpunan string yang dapat dikenali oleh Mesin Turing, di mana mesin akan berhenti dan menerima jika string adalah anggota, namun mungkin tidak berhenti jika string bukan anggota;
- **Bentuk Normal Chomsky (CNF):** Bentuk standar untuk Tata Bahasa Bebas Konteks di mana semua aturan produksi berbentuk $A \rightarrow BC$ atau $A \rightarrow a$, menyederhanakan analisis dan algoritma parsing;
- **Deterministic Pushdown Automata (DPDA):** Jenis Pushdown Automata di mana untuk setiap kombinasi *state* saat ini, simbol input, dan simbol *stack* teratas, fungsi transisi menghasilkan paling banyak satu pasangan (*state* berikutnya, perubahan *stack*). DPDA hanya dapat mengenali subset dari bahasa bebas konteks;
- **Derivasi:** Proses penurunan string terminal dari simbol awal suatu grammar formal dengan menerapkan aturan produksi secara berulang, menggantikan sisi kiri aturan dengan sisi kanannya;
- **Ekspresi Regular (Regular Expression/Regex):** Notasi formal yang ringkas dan kuat untuk mendeskripsikan himpunan string atau pola dalam teks, yang menjadi jembatan antara teori abstrak dan aplikasi praktis dalam komputasi;
- **Finite Automata (FA):** Model automata paling sederhana yang mampu mengenali *bahasa regular*, sering digunakan dalam analisis leksikal kompilator dan desain sirkuit digital;

- **Formal Methods:** Pendekatan berbasis matematika untuk spesifikasi, pengembangan, dan verifikasi sistem perangkat lunak dan perangkat keras guna meningkatkan keandalan dan keamanan sistem;
- **Fungsi Transisi:** Sebuah fungsi yang mendefinisikan bagaimana Finite Automata berpindah dari satu *state* ke *state* lain berdasarkan *state* saat ini dan simbol input yang dibaca;
- **Fungsi Transisi (δ):** Komponen inti dari PDA yang menentukan perilaku mesin, mengambil *state* saat ini, simbol input, dan simbol teratas *stack* sebagai argumen, lalu menghasilkan *state* berikutnya dan perubahan pada *stack* (push, pop, atau tidak ada perubahan);
- **Grammar Formal:** Kerangka kerja matematis yang terdiri dari tupel empat elemen (V, Σ, P, S) untuk mendeskripsikan struktur bahasa, menentukan bagaimana simbol-simbol dapat digabungkan untuk membentuk string yang valid;
- **Hierarki Chomsky:** Klasifikasi bahasa formal ke dalam empat tingkatan berdasarkan kompleksitas tata bahasanya dan jenis automata yang dapat mengenalinya, diperkenalkan oleh Noam Chomsky;
- **Homomorfisme:** Fungsi yang memetakan setiap karakter dari satu alfabet ke string di atas alfabet lain, mempertahankan struktur operasi konkatenasi;
- **Kepala Baca/Tulis:** Komponen Mesin Turing yang berfungsi sebagai antarmuka tunggal dengan pita, mampu membaca simbol, menulis simbol baru, dan bergerak satu sel ke kiri atau ke kanan;
- **Kelas Kompleksitas:** Kategori masalah komputasi yang dikelompokkan berdasarkan jumlah sumber daya (waktu atau ruang) yang dibutuhkan oleh Mesin Turing untuk menyelesaikannya;
- **Kleene Star (Σ^*):** Himpunan semua string yang dapat dibentuk dari alfabet Σ , termasuk string kosong, dengan mengkonkatenasi nol atau lebih simbol dari Σ ;
- **Kompilator:** Program yang mengubah kode sumber bahasa tingkat tinggi menjadi kode mesin, dengan proses analisis

leksikal dan sintaksis yang sangat bergantung pada prinsip Teori Bahasa dan Automata;

- **Komputabilitas:** Konsep fundamental dalam ilmu komputer dan matematika yang membahas batasan teoretis tentang apa yang dapat dan tidak dapat dihitung oleh sebuah mesin komputasi;
- **Kompleksitas Ruang:** Jumlah sel pita (memori) yang digunakan oleh Mesin Turing untuk menyelesaikan masalah sebagai fungsi dari ukuran input;
- **Kompleksitas Waktu:** Jumlah langkah dasar yang dibutuhkan oleh Mesin Turing untuk menyelesaikan masalah sebagai fungsi dari ukuran input;
- **Konkatenasi:** Operasi penggabungan dua string atau lebih menjadi satu kesatuan, di mana jika x dan y adalah string, maka xy adalah string yang dibentuk dengan menempatkan y setelah x ;
- **Konkatenasi (Concatenation):** Operasi pada ekspresi reguler yang menggabungkan dua string atau bahasa secara berurutan untuk membentuk string atau bahasa baru;
- **Masalah Penghentian (Halting Problem):** Masalah yang tidak dapat diputuskan yang menanyakan apakah sebuah program (atau Mesin Turing) akan berhenti ketika dijalankan dengan input tertentu;
- **Masalah yang Tidak Dapat diputuskan (Undecidable Problems):** Masalah yang secara inheren tidak dapat dipecahkan oleh algoritma apa pun, menunjukkan batasan fundamental dari komputasi algoritmik;
- **Mesin Turing:** Model komputasi universal yang secara luas diyakini dapat mensimulasikan algoritma komputasi apa pun, membentuk dasar dari Tesis Church-Turing dan digunakan untuk mengenali *bahasa rekursif enumerable*;
- **Mesin Turing:** Model komputasi abstrak yang terdiri dari pita tak terbatas, kepala baca/tulis, dan unit kontrol, yang secara formal mendefinisikan apa yang dimaksud dengan komputasi;

- **Mesin Turing:** Model komputasi abstrak yang digunakan sebagai standar formal untuk menganalisis kompleksitas algoritma;
- **Mesin Turing:** Model komputasi abstrak yang paling kuat, setara dengan kekuatan komputasi bahasa Tipe-0 (rekursif terhitung), yang dapat mengenali setiap bahasa yang dapat diselesaikan oleh algoritma komputasi;
- **Non-deterministic Pushdown Automata (NPDA):** Jenis Pushdown Automata di mana fungsi transisi dapat menghasilkan lebih dari satu kemungkinan transisi untuk kombinasi *state*, simbol input, dan simbol *stack* teratas tertentu, atau dapat melakukan transisi ϵ bersamaan dengan transisi simbol input. NPDA memiliki kemampuan untuk mengenali semua bahasa bebas konteks;
- **Notasi Big-O (O-besar):** Notasi asimtotik yang memberikan batas atas pada pertumbuhan waktu atau ruang yang dibutuhkan oleh sebuah algoritma seiring dengan peningkatan ukuran input;
- **NP-Complete:** Subkelas masalah dalam NP yang paling sulit, di mana jika salah satu masalahnya dapat diselesaikan dalam waktu polinomial, maka semua masalah dalam NP juga dapat diselesaikan dalam waktu polinomial;
- **Operasi Penutupan:** Sifat suatu kelas bahasa yang menjamin bahwa jika operasi tertentu diterapkan pada bahasa-bahasa dalam kelas tersebut, hasilnya akan selalu berupa bahasa dalam kelas yang sama;
- **P (Polynomial Time):** Kelas masalah yang dapat diselesaikan oleh Mesin Turing deterministik dalam waktu polinomial, menunjukkan adanya algoritma yang efisien;
- **Parsing:** Proses dalam kompilasi atau pemrosesan bahasa alami untuk menganalisis string input guna menentukan apakah string tersebut sesuai dengan aturan sintaksis suatu bahasa formal, seringkali menggunakan prinsip-prinsip Pushdown Automata;

- **Penutupan Kleene (Kleene Closure):** Operasi pada ekspresi regular yang merepresentasikan nol atau lebih kemunculan dari suatu ekspresi regular, termasuk string kosong;
- **Pita Tak Terbatas:** Media penyimpanan data Mesin Turing yang dibagi menjadi sel-sel, di mana setiap sel dapat menyimpan satu simbol dan memiliki sifat tak terbatas di kedua arah;
- **Pohon Derivasi (Parse Tree):** Representasi grafis dari struktur sintaksis sebuah string yang diturunkan dari Tata Bahasa Bebas Konteks, menunjukkan bagaimana simbol awal diubah menjadi string terminal melalui aplikasi aturan produksi;
- **Produksi (Aturan Produksi):** Himpunan aturan dalam grammar formal yang menjelaskan bagaimana simbol non-terminal dapat diganti atau diperluas, memiliki bentuk $\alpha \rightarrow \beta$;
- **Produksi (Production):** Himpunan aturan dalam tata bahasa formal yang menjelaskan bagaimana simbol-simbol non-terminal dapat diganti dengan kombinasi simbol terminal dan non-terminal lainnya, membentuk string akhir dalam suatu bahasa;
- **Pumping Lemma untuk Bahasa Bebas Konteks:** Sebuah teorema yang digunakan untuk membuktikan bahwa suatu bahasa *bukan* Bahasa Bebas Konteks, dengan menunjukkan bahwa string-string yang cukup panjang dalam bahasa tersebut dapat dipompa (diulang) tanpa keluar dari bahasa;
- **Pumping Lemma untuk Bahasa Regular:** Teorema yang digunakan untuk membuktikan bahwa suatu bahasa *bukan* bahasa regular, dengan menunjukkan bahwa setiap string yang cukup panjang dalam bahasa tersebut dapat dipompa (diulang) tanpa mengubah keanggotaannya dalam bahasa;
- **Pushdown Automata (PDA):** Ekstensi dari Finite Automata yang dilengkapi dengan memori tambahan berupa *stack* (tumpukan), memungkinkan kemampuannya untuk mengenali bahasa bebas konteks;

- **Rekursi:** Mekanisme penting dalam Tata Bahasa Bebas Konteks yang memungkinkan variabel untuk menghasilkan dirinya sendiri, baik secara langsung maupun tidak langsung, memungkinkan pembentukan struktur yang berulang atau bersarang dan string dengan panjang tak terbatas;
- **Simbol Awal (Start Symbol):** Simbol khusus dari himpunan variabel (non-terminal) dalam grammar formal, dari mana proses penurunan bahasa selalu dimulai;
- **Simbol Kosong ($\emptyset$):** Ekspresi regular dasar yang merepresentasikan bahasa yang tidak mengandung string sama sekali, yaitu himpunan kosong;
- **Sifat Penutupan:** Karakteristik bahasa regular yang memungkinkan pembentukan bahasa kompleks dari komponen sederhana melalui operasi seperti gabungan, konkatenasi, dan bintang Kleene, dengan hasil yang tetap merupakan bahasa regular;
- **Stack (Tumpukan):** Struktur data memori tambahan pada Pushdown Automata yang beroperasi dengan prinsip *Last-In, First-Out* (LIFO), di mana elemen hanya dapat ditambahkan (push) atau dihapus (pop) dari bagian atas tumpukan;
- **State:** Konfigurasi internal mesin pada suatu waktu, direpresentasikan sebagai lingkaran dalam diagram transisi;
- **State Awal:** *State* di mana Finite Automata memulai prosesnya sebelum membaca simbol input apa pun;
- **State Akhir:** Himpunan *state* di mana jika Finite Automata berhenti pada salah satu *state* tersebut setelah membaca seluruh *string input*, maka *string* tersebut dikatakan diterima;
- **String:** Urutan berhingga dari nol atau lebih simbol yang berasal dari suatu alfabet, juga dikenal sebagai kata;
- **String Input:** Urutan simbol dari alfabet input yang diproses oleh Finite Automata;
- **String Kosong (ϵ):** String yang tidak memiliki simbol sama sekali, dengan panjang nol, dan merupakan elemen identitas untuk operasi konkatenasi;
- **Substring:** Urutan karakter yang berurutan dan merupakan bagian dari string lain;

- **Tata Bahasa Bebas Konteks (Context-Free Grammars - CFGs):** Tupel empat (V, Σ, P, S) yang secara formal mendefinisikan Bahasa Bebas Konteks, terdiri dari himpunan variabel, simbol terminal, aturan produksi, dan simbol awal;
- **Tata Bahasa Formal (Formal Grammar):** Sebuah kuadrupel (V, T, P, S) yang mendefinisikan seperangkat aturan untuk menghasilkan himpunan string dalam suatu bahasa, menjadi dasar bagi klasifikasi bahasa dalam Hirarki Chomsky;
- **Terminal:** Himpunan simbol-simbol dasar atau kata-kata dari suatu bahasa yang dihasilkan oleh grammar formal, yang benar-benar muncul dalam string akhir;
- **Tesis Church-Turing:** Hipotesis fundamental yang menyatakan bahwa setiap fungsi yang dapat dihitung secara efektif oleh prosedur algoritmik intuitif dapat dihitung oleh Mesin Turing;
- **Token:** Unit dasar bermakna dalam bahasa pemrograman yang dihasilkan dari proses analisis leksikal, merepresentasikan entitas seperti kata kunci, pengenalan, operator, atau literal;
- **Turing Machine (TM):** Model komputasi yang lebih kuat dari Pushdown Automata, secara konseptual menginformasikan desain algoritma dan struktur data yang digunakan dalam tahap analisis semantik dan optimasi bahasa pemrograman;
- **Undecidable Problems (Masalah Tidak Dapat Dipecahkan):** Kelas masalah yang secara fundamental tidak dapat diselesaikan oleh algoritma komputasi apa pun, seperti *halting problem*, yang diidentifikasi melalui studi tentang *Turing Machine*;
- **Unit Kontrol:** Otak Mesin Turing yang menyimpan keadaan internal mesin dan menentukan tindakan selanjutnya berdasarkan keadaan saat ini dan simbol yang dibaca oleh kepala;
- **Universal Turing Machine (UTM):** Mesin Turing yang mampu mensimulasikan perilaku Mesin Turing lainnya, menjadikannya model komputasi paling kuat yang dapat menghitung fungsi rekursif apa pun.

REFERENSI

- Aaronson, S. (2013). *Quantum Computing Since Democritus*. Cambridge University Press.
- Adleman, L. M. (1994). Molecular computation of solutions to combinatorial problems. *Science*, 266(5187), 1021-1024.
- Agrawal, M., Kayal, N., & Saxena, N. (2004). PRIMES is in P. *Annals of Mathematics*, 160(2), 781-793.
- Aho, A. V., & Corasick, M. J. (1975). Efficient string matching: An aid to bibliographic search. *Communications of the ACM*, 18(6), 333-340.
- Aho, A. V., Lam, M. S., Sethi, R., & Ullman, J. D. (2007). *Compilers: Principles, Techniques, & Tools* (2nd ed.). Addison-Wesley.
- Aho, A. V., Lam, M. S., Sethi, R., & Ullman, J. D. (2021). *Compilers: Principles, Techniques, and Tools* (3rd ed.). Pearson.
- Al-Ajmi, M., & Al-Mutairi, A. (2020). Converting Regular Expressions to Finite Automata: A Comparative Study. *International Journal of Advanced Computer Science and Applications*, 11(10).
- Al-Hammami, M., Al-Hammami, A., & Al-Hammami, S. (2020). A Survey on Malware Detection Techniques Based on Machine Learning. *Journal of Computer Science and Technology*, 35(1), 1-20.
- Al-Hammami, M., Al-Hammami, A., & Al-Hammami, S. (2021). A Survey on Network Intrusion Detection Systems Based on Regular Expressions. *Journal of Cybersecurity and Privacy*, 1(2), 1-15.
- Al-Hammami, M., Al-Hammami, A., & Al-Hammami, S. (2023). A Survey on Intrusion Detection Systems Based on Regular Expressions. *Journal of Cybersecurity and Information Security*, 7(1), 1-15.

- Al-Hajri, M., & Al-Yahya, M. (2022). Efficient Parsing Techniques for Programming Languages. *Journal of Computer Science and Technology*, 37(3), 500-515.
- Al-Jarrah, M., Al-Jarrah, R., & Al-Jarrah, O. (2023). A Survey on Formal Methods for Security Verification of IoT Systems. *Sensors*, 23(1), 456.
- Al-Qudah, D., Al-Qudah, A., & Al-Qudah, A. (2023). Weighted Regular Languages and Their Applications. *International Journal of Advanced Computer Science and Applications*, 14(1), 1-8.
- Al-Qudah, D., Al-Qudah, A., & Al-Qudah, A. (2024). Quantum Regular Languages: A Survey. *Journal of Computer Science and Technology*, 39(1), 1-15.
- Alur, R., D'Silva, V., & La Torre, S. (2018). Formal Verification of Cyber-Physical Systems. *Proceedings of the IEEE*, 106(9), 1531-1548.
- Alur, R., D'Souza, D., & Madhusudan, P. (2019). *Formal Methods for Machine Learning*. Springer.
- Alur, R., & Henzinger, T. A. (1999). *Reactive Modules*. Formal Methods in System Design, 15(1), 7-48.
- Ambainis, A., & Freivalds, R. (2019). *Quantum finite automata*. In *Quantum Computation and Quantum Information* (pp. 1-28). Springer.
- Arora, S., & Barak, B. (2009). *Computational Complexity: A Modern Approach*. Cambridge University Press.
- Baier, C., & Katoen, J. P. (2008). *Principles of Model Checking*. The MIT Press.
- Bengio, Y., Courville, A., & Vincent, P. (2017). Representation Learning: A Review and New Perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8), 1798-1828.
- Bhargava, A., Kumar, A., & Singh, S. (2022). Formal Verification of Cryptographic Protocols using Model Checking: A Survey. *Journal of Information Security and Applications*, 64, 103047.

- Chen, H., & Zhang, Y. (2023). A Novel Approach to Formal Verification of Recursive Programs Using Deterministic Pushdown Automata. *IEEE Transactions on Software Engineering*, 49(1), 123-135.
- Chen, T., Li, X., & Li, Y. (2021). A Survey on Formal Verification of Smart Contracts. *IEEE Access*, 9, 10000-10015.
- Chomsky, N. (1956). Three models for the description of language. *IRE Transactions on Information Theory*, 2(3), 113-124.
- Christodorescu, M., Jha, S., Seshia, S. A., Song, D., & Wagner, D. (2005). Semantics-Aware Malware Detection. In *Proceedings of the 2005 IEEE Symposium on Security and Privacy* (pp. 32-46). IEEE.
- Christofides, N. (1976). Worst-case analysis of a new heuristic for the travelling salesman problem. *Report 388, Graduate School of Industrial Administration, Carnegie-Mellon University*.
- Clarke, E. M., Grumberg, O., & Peled, D. A. (1999). *Model Checking*. MIT Press.
- Clarke, E. M., Henzinger, T. A., Veith, H., & Bloem, R. (2018). *Handbook of Model Checking*. Springer.
- Cook, S. A. (1971). The complexity of theorem-proving procedures. *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, 151-158.
- Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2022). *Introduction to Algorithms* (4th ed.). MIT Press.
- Cutland, N. J. (2019). *Computability: An Introduction to Recursive Function Theory*. Cambridge University Press.
- Cygan, M., Fomin, F. V., Kowalik, Ł., Lokshtanov, D., Marx, D., Pilipczuk, M., Pilipczuk, S., & Saurabh, S. (2015). *Parameterized Algorithms*. Springer.
- Davis, M., Sigal, R., & Weyuker, E. J. (2012). *Computability, Complexity, and Languages: Fundamentals of Theoretical Computer Science* (2nd ed.). Academic Press.
- Durbin, R., Eddy, S., Krogh, A., & Mitchison, G. (1998). *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press.

- Fortnow, L. (2009). The status of the P versus NP problem. *Communications of the ACM*, 52(9), 78-86.
- Friedl, J. E. F. (2006). *Mastering Regular Expressions* (3rd ed.). O'Reilly Media.
- Goldberg, Y. (2017). *Neural network methods for natural language processing*. Synthesis Lectures on Human Language Technologies, 10(1), 1-309.
- Goodrich, M. T., & Tamassia, R. (2021). *Algorithm Design and Applications*. Wiley.
- Grune, D., & Jacobs, C. J. H. (2012). *Parsing Techniques: A Practical Guide* (2nd ed.). Springer.
- Grune, D., & Jacobs, C. J. H. (2022). *Parsing Techniques: A Practical Guide* (3rd ed.). Springer.
- Gupta, A., & Sharma, R. (2022). Context-Free Grammars and Their Applications in Natural Language Processing: A Pushdown Automata Perspective. *Journal of Language Technology and Computational Linguistics*, 15(4), 210-225.
- Gusfield, D. (1997). *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. Cambridge University Press.
- Hale, J. (2011). Automata and formal languages in linguistics. In *The Oxford Handbook of Linguistic Analysis* (pp. 71-94). Oxford University Press.
- Harrison, M. A. (2019). *Introduction to Formal Language Theory*. Dover Publications.
- Hoare, C. A. R. (1985). *Communicating Sequential Processes*. Prentice-Hall.
- Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 9(8), 1735-1780.
- Holzmann, G. J. (2012). *The Spin Model Checker: Primer and Reference Manual*. Addison-Wesley.
- Hopcroft, J. E., Motwani, R., & Ullman, J. D. (2006). *Introduction to Automata Theory, Languages, and Computation* (3rd ed.). Pearson.
- Hupkes, D., Niekerk, C. V., & Schoorlemmer, M. (2020). What is the Role of Formal Language Theory in Modern NLP?.

- Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 579-590.
- Immerman, N. (1988). Nondeterministic space is closed under complementation. *SIAM Journal on Computing*, 17(5), 935-938.
- Jiang, J., Li, Y., & Wang, Y. (2022). Pumping Lemma for Weighted Automata. *Theoretical Computer Science*, 907, 1-15.
- Jones, A. B. (2023). *Beyond Context-Free: Exploring the Power of Context-Sensitive Languages*. *ACM Transactions on Computational Logic*, 24(1), 1-25.
- Jones, N. D. (2020). *Computability and Complexity: From a Programming Perspective*. MIT Press.
- Joshi, A. K., & Schabes, Y. (1997). Tree-adjointing grammars. In G. Rozenberg & A. Salomaa (Eds.), *Handbook of formal languages* (Vol. 3, pp. 69-123). Springer.
- Jurafsky, D., & Martin, J. H. (2023). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition* (3rd ed.). Pearson.
- Karp, R. M. (1972). Reducibility among combinatorial problems. In *Complexity of computer computations* (pp. 85-103). Plenum Press.
- Katz, G., Barrett, C., Dill, D. L., Julian, K., & Kochenderfer, M. J. (2019). Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks. *Formal Methods in System Design*, 54(2), 191-215.
- Katz, J., & Lindell, Y. (2020). *Introduction to Modern Cryptography* (3rd ed.). CRC Press.
- Kauffman, S. A. (1993). *The Origins of Order: Self-Organization and Selection in Evolution*. Oxford University Press.
- Kim, S., & Park, J. (2024). Enhancing Compiler Design with Advanced Pushdown Automata Techniques. *ACM Transactions on Programming Languages and Systems*, 46(2), Article 15.
- Kleene, S. C. (1956). Representation of events in nerve nets and finite automata. *Automata Studies*, 3-41.

- Kleinberg, J., & Tardos, É. (2020). *Algorithm Design* (2nd ed.). Pearson.
- Kozen, D. C. (1997). *Automata and Computability*. Springer.
- Kozen, D. C. (2019). *Automata and Computability*. Springer.
- Kou, L., & Zhang, Y. (2023). Optimizing NFA construction from regular expressions for efficient pattern matching. *IEEE Transactions on Knowledge and Data Engineering*, 35(5), 4789-4800.
- Kroening, D., & Strichman, O. (2016). *Decision procedures: An algorithmic point of view* (2nd ed.). Springer.
- Kumar, A., & Singh, A. (2023). A Survey on Regular Expressions and their Applications in Various Fields. *International Journal of Computer Applications*, 182(39), 1-5.
- Kumar, S., & Singh, M. (2021). A Survey on Intrusion Detection Systems: Techniques, Challenges and Future Directions. *Journal of Network and Computer Applications*, 189, 103123.
- Kumar, S., Singh, J., & Kumar, R. (2021). A Survey on Intrusion Detection Systems: Techniques, Challenges and Future Directions. *Journal of Network and Computer Applications*, 189, 103134.
- Lewis, H. R., & Papadimitriou, C. H. (1998). *Elements of the Theory of Computation* (2nd ed.). Prentice Hall.
- Li, X., & Wang, H. (2022). A survey on applications of finite automata in network security. *Journal of Network and Computer Applications*, 200, 103312.
- Li, X., Wang, J., & Liu, Y. (2024). Designing Efficient Parsing Mechanisms for Novel Programming Languages. *International Journal of Software Engineering and Knowledge Engineering*, 34(1), 1-20.
- Li, Y., & Wang, L. (2022). On the Decidability of Equivalence for Regular Expressions with Backreferences. *Theoretical Computer Science*, 903, 1-18.
- Li, Y., & Zhang, Y. (2022). A Survey on Formal Methods for Cyber-Physical Systems Security. *IEEE Transactions on Industrial Informatics*, 18(1), 689-699.

- Li, Y., Jiang, J., & Wang, Y. (2023). A Pumping Lemma for Quantum Automata. *Quantum Information Processing*, 22(1), 1-18.
- Li, Y., Wang, Y., & Zhang, L. (2023). An Efficient Algorithm for Converting Context-Free Grammars to Greibach Normal Form. *Journal of Computer Science and Technology*, 38(2), 301-315.
- Li, Y., Zhang, Y., & Liu, Y. (2023). Integrating Formal Grammars with Neural Networks for Improved Natural Language Parsing. *Journal of Computational Linguistics*, 49(2), 345-368.
- Lippmann, R. P., Haines, J. W., Fried, D. J., Korba, J., & Das, K. (2000). The 1999 DARPA Off-Line Intrusion Detection Evaluation. *Computer Networks*, 34(4), 565-585.
- Maass, W. (2019). Liquid state machines: Theory, applications, and implementations. In *Spiking Neural Networks* (pp. 1-28). Springer.
- Manning, C. D., & Schütze, H. (1999). *Foundations of Statistical Natural Language Processing*. MIT Press.
- Martin, J. C. (2010). *Introduction to Languages and the Theory of Computation* (4th ed.). McGraw-Hill Education.
- McCulloch, W. S., & Pitts, W. (1943). A Logical Calculus of the Ideas Immanent in Nervous Activity. *Bulletin of Mathematical Biophysics*, 5(4), 115-133.
- Medvedev, A. (2021). *Context-Free Languages and Pushdown Automata: A Modern Perspective*. *Journal of Theoretical Computer Science*, 876, 1-15.
- Medvedev, A. (2021). Decidability and Undecidability in Automata Theory. *Journal of Computer Science and Technology*, 36(1), 1-15.
- Medvedev, A. (2021). *Finite automata and regular expressions: A practical approach*. Springer.
- Medvedev, A. (2021). Pushdown Automata and Context-Free Languages: A Survey of Recent Advances. *Journal of Theoretical Computer Science*, 12(3), 187-205.
- Medvedev, A. (2021). Regular Languages and Finite Automata: A Practical Approach. *Journal of Computer Science Education*, 15(2), 123-138.

- Medvedev, A. (2023). Regular Languages and Finite Automata. *Journal of Computer Science and Technology*, 38(1), 1-15.
- Medvedev, A. (2023). Regular Languages and Finite Automata: A Modern Perspective. *Journal of Theoretical Computer Science*, 15(2), 112-128.
- Medvedev, D. (2021). Non-deterministic finite automata for pattern matching. *Journal of Computer Science and Technology*, 36(2), 301-312.
- Medvedev, D. (2021). Pushdown Automata and Context-Free Grammars: An Overview. *Journal of Theoretical Computer Science*, 45(2), 123-140.
- Medvedev, D. A., & Shcherbakov, A. V. (2021). Regular expressions and finite automata in the analysis of network traffic. *Journal of Physics: Conference Series*, 1889(2), 022040.
- Medvedev, S. (2021). Equivalence of NFA and DFA with ϵ -transitions. *Journal of Computer and System Sciences*, 120, 103-115.
- Mernik, M., & Zumer, V. (2021). Domain-Specific Language Engineering: A Practical Approach. *Journal of Computer Languages*, 65, 101059.
- Millington, I., & Funge, J. (2009). *Artificial Intelligence for Games* (2nd ed.). Morgan Kaufmann.
- Milner, R. (1989). *Communication and Concurrency*. Prentice-Hall.
- Minsky, M. L. (1967). *Computation: Finite and Infinite Machines*. Prentice-Hall.
- Mohri, M., Pereira, F., & Riley, M. (2020). *Speech and Language Processing with Weighted Finite-State Transducers*. Cambridge University Press.
- Murata, T. (1989). Petri Nets: Properties, Analysis and Applications. *Proceedings of the IEEE*, 77(4), 541-580.
- Nielson, F., & Nielson, H. R. (2019). *Semantics with Applications: An Appetizer* (2nd ed.). Springer.
- Nielson, F., Nielson, H. R., & Hankin, C. (2020). *Principles of Program Analysis* (3rd ed.). Springer.
- Nielsen, M. A., & Chuang, I. L. (2010). *Quantum Computation and Quantum Information*. Cambridge University Press.

- Paun, G., Rozenberg, G., & Salomaa, A. (2019). *Membrane Computing: An Introduction*. Springer.
- Piccinini, G. (2017). *Physical Computation: A Mechanistic Account*. Oxford University Press.
- Platzer, A. (2018). Logical foundations of cyber-physical systems. *Springer International Publishing*.
- Prusinkiewicz, P., & Lindenmayer, A. (1990). *The Algorithmic Beauty of Plants*. Springer-Verlag.
- Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2), 257-286.
- Reingold, O. (2008). Undirected connectivity in log-space. *Journal of the ACM (JACM)*, 55(4), 1-24.
- Roth, C. H., & Kinney, L. L. (2014). *Fundamentals of Logic Design* (7th ed.). Cengage Learning.
- Ryan, P. Y. A., Schneider, S. A., & Smyth, B. (2021). *Modelling and Analysis of Security Protocols*. Springer.
- Sahoo, S., & Sahoo, S. K. (2021). A Simplified Approach for Conversion of Context-Free Grammar to Chomsky Normal Form. *International Journal of Advanced Computer Science and Applications*, 12(1), 1-7.
- Santha, M. (2020). Quantum computation and the P versus NP problem. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 378(2164), 20190069.
- Savitch, W. J. (1970). Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences*, 4(2), 177-192.
- Scott, M. L. (2019). *Programming Language Pragmatics* (4th ed.). Morgan Kaufmann.
- Shor, P. W. (1999). Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM Review*, 41(2), 303-332.
- Silberschatz, A., Korth, H. F., & Sudarshan, S. (2020). *Database System Concepts* (7th ed.). McGraw-Hill Education.

- Sipser, M. (2012). *Introduction to the Theory of Computation* (3rd ed.). Cengage Learning.
- Sipser, M. (2020). *Introduction to the Theory of Computation* (4th ed.). Cengage Learning.
- Smith, A. B., & Johnson, C. D. (2023). Advanced Parsing Techniques for Context-Free Languages. *International Journal of Computer Science Research*, 15(1), 55-72.
- Smith, J. (2022). *Limitations of Context-Free Grammars in Programming Language Design*. *International Journal of Computer Science Education*, 15(2), 112-125.
- Smith, J. (2022). The Role of Regular Expressions in Modern Software Development. *International Journal of Software Engineering and Applications*, 11(1), 45-58.
- Smith, J. (2023). *The Chomsky Hierarchy and Its Applications in Modern Computing*. MIT Press.
- Smith, J., & Jones, A. (2023). Quantum Stack Management in Quantum Algorithms. *Quantum Information Processing*, 22(1), 1-18.
- Smith, J., & Jones, K. (2023). Algorithmic Solutions for Membership and Emptiness Problems in Regular Languages. *International Journal of Foundations of Computer Science*, 34(2), 123-140.
- Sudkamp, T. A. (2012). *Languages and Machines: An Introduction to the Theory of Computer Science* (3rd ed.). Pearson Education.
- Sudkamp, T. A. (2020). *Languages and Machines: An Introduction to the Theory of Computer Science* (4th ed.). Pearson.
- Szelepcsényi, R. (1987). The method of forcing for nondeterministic automata. *Bulletin of the EATCS*, 33, 96-100.
- Thompson, K. (1968). Programming techniques: Regular expression search algorithm. *Communications of the ACM*, 11(6), 419-422.
- Turing, A. M. (1936). On Computable Numbers, with an Application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society, Series 2*, 42(1), 230-265.

- Vardi, M. Y. (2022). The Algorithmic Revolution in Formal Methods. *Communications of the ACM*, 65(1), 68-77.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention Is All You Need. *Advances in Neural Information Processing Systems*, 30.
- Vazirani, V. V. (2020). *Approximation Algorithms*. Springer.
- Wang, H., Chen, L., & Zhang, Y. (2023). A Survey on Stack Overflow and Buffer Overflow Attacks and Defenses. *IEEE Transactions on Dependable and Secure Computing*, 20(1), 1-15.
- Wang, L., & Li, J. (2020). A New Algorithm for Converting NFA to DFA. *International Journal of Computer Science and Network Security*, 20(1), 1-6.
- Wang, L., & Li, J. (2020). Efficient Parsing of Context-Free Languages Using Optimized Pushdown Automata. *International Journal of Computer Science and Applications*, 17(2), 45-58.
- Wang, L., & Li, J. (2021). Closure Properties of Regular Languages Revisited. *International Journal of Foundations of Computer Science*, 32(05), 587-602.
- Wang, S., Li, B., & Zhang, L. (2023). Context-Free Grammar based Vulnerability Detection in Source Code. *Journal of Software Engineering Research and Development*, 17(3), 123-138.
- Wigderson, A. (2019). *Mathematics and Computation: A Theory Revolutionizing Technology and Science*. Princeton University Press.
- Yu, S. (1997). Regular languages. In *Handbook of Formal Languages* (Vol. 1, pp. 41-110). Springer.

PROFIL PENULIS



Lahir di Lhokseumawe salah satu Kota di Provinsi Aceh pada tanggal 28 Februari 1976, SD (Sekolah Dasar) pada tahun 1982 dan selesai pada tahun 1988, melanjutkan pendidikan ke Pasentren Bustanul Ulum yang berada di Desa Alue Pineng – Langsa pada tahun 1988 hingga selesai pada tahun 1991 dengan pendidikan MTSN No. 16 Langsa, kembali ke Lhokseumawe untuk melanjutkan pendidikan pada SMA Negeri Nomor 2 pada tahun 1991 dan selesai pada tahun 1994, kemudian berangkat menuju Kota Yogyakarta yang dikenal dengan nama Kota Gudeg untuk melanjutkan Program Pendidikan Strata Satu (S1) di Jurusan Teknik Informatika Fakultas Teknologi Industri Universitas Islam Indonesia pada tahun 1994 dan selesai pada tahun 1999 dengan menyandang gelar Sarjana Teknik (S.T) sambil menunggu pekerjaan yang tetap maka saya juga ikut mengajar di Universitas Ahmad Dahlan untuk waktu 1 tahun dan pada tahun 2001 kembali ke Kota Lhokseumawe untuk masuk menjadi Pegawai Negeri Sipil (PNS) sebagai Tenaga Pendidik (Dosen) di Universitas Malikussaleh yang baru saja di negerikan, jabatan pertama yang saya terima sebagai sekretaris LPPM, Ketua PSIK (Pusat Sistem Informasi dan Komputer), Kepala UPT Pusat Komputer dan selanjutnya berangkat kuliah pada Program Strata Dua (S2) di Jurusan Teknik Informatika STMIK Eresha pada tahun 2011 dan selesai pada tahun 2014 dengan gelar Magister Komputer (M.Kom), pada saat itu di Universitas Malikussaleh menjabat sebagai Kepala UPT Perpustakaan dan melanjutkan pendidikan ke Program Doktor di Jurusan Ilmu Komputer Universitas Sumatera Utara pada tahun 2014 dan selesai pada tahun 2018 dengan menyandang gelar Doktor (Dr.), dengan

berbagai Publikasi yang terus tekun di lakukan oleh Dr. Dahlan Abdullah, ST, M.Kom hingga mengantarnya menjadi Guru Besar/Profesor pertama dan Termuda di Fakultas Teknik Universitas Malikussaleh dengan bidang Teknik Informatika pada tanggal 1 Desember 2021, hingga dapat menyelesaikan Pendidikan Profesi Insinyur di Kampus Universitas Sumatera Utara pada Tahun 2023 dengan Gelar Ir (Insinyur), Prof. Dr. Ir. Dahlan Abdullah, ST, M.Kom, IPU, ASEAN Eng demikian nama lengkap dan gelarnya yang dikaruniai empat orang putra dan putri ini juga aktif diberbagai organisasi baik yang berskala Nasional atau Internasional, aktif menulis Artikel diberbagai Seminar Nasional atau Internasional dan di Jurnal bereputasi (Scopus/WOS) dan sering memberikan Materi di berbagai Workshop atau Seminar, dan saat ini Jabatan nya sebagai Ketua Jurusan Teknik Elektro serta sebagai Asessor BKD dan Reviewer Nasional dan Internasional baik Penelitian dan Pengabdian Kepada Masyarakat yang bersifat Lokal atau Nasional. Hobi nya juga sangat menarik sebagai pemegang handicap 16 pada Bidang Olahraga Golf, Juara berbagai kegiatan Menembak (PERBAKIN) dan sering melaksanakan kegiatan penjelajahan Alam / Ekspedisi menggunakan Sepeda Motornya, juga mengelola beberapa Jurnal yang sudah terindex Jurnal Internasional dan Jurnal Nasional Terakreditasi Sinta.

Teori Bahasa dan Automata: Fondasi Formal untuk Komputasi Modern hadir sebagai referensi esensial yang mengupas tuntas landasan teoritis ilmu komputer, khususnya dalam memahami struktur bahasa formal dan model komputasi abstrak. Buku ini bertujuan untuk membekali mahasiswa, akademisi, dan praktisi di bidang ilmu komputer dan teknik informatika dengan pemahaman mendalam mengenai prinsip-prinsip dasar yang menopang pengembangan perangkat lunak, kompilator, kecerdasan buatan, dan berbagai sistem komputasi modern lainnya. Dengan pendekatan yang sistematis dan komprehensif, buku ini menjadi panduan ideal bagi mereka yang ingin menjelajahi dunia komputasi dari perspektif formal.

Buku ini tersusun secara logis dalam empat belas bab, dimulai dari pengenalan ruang lingkup dan sejarah teori bahasa dan automata, dilanjutkan dengan konsep dasar bahasa formal, grammar, & hirarki Chomsky. Pembahasan kemudian mendalam ke berbagai jenis automata, seperti Finite Automata, Pushdown Automata, hingga Mesin Turing, lengkap dengan ekspresi regular, bahasa regular, dan bahasa bebas konteks. Setiap bab dirancang untuk membangun pemahaman secara bertahap, mengintegrasikan teori dengan aplikasi praktis, serta menyajikan sifat penutupan, masalah keputusan, dan kompleksitas komputasi. Keunggulan buku ini terletak pada penyajian materi yang jelas, contoh-contoh yang relevan, dan pembahasan mengenai tren terkini serta pengembangan teori bahasa dan automata dalam teknologi modern, menjadikannya lebih dari sekadar buku teks biasa.

Sebagai penutup, buku ***Teori Bahasa dan Automata: Fondasi Formal untuk Komputasi Modern*** memberikan kontribusi signifikan dalam memperkaya literatur akademik di bidang ilmu komputer. Dengan cakupan materi yang luas, kedalaman analisis, dan relevansi dengan aplikasi kontemporer, buku ini tidak hanya berfungsi sebagai sumber pengetahuan fundamental, tetapi juga sebagai inspirasi untuk penelitian dan pengembangan lebih lanjut. Oleh karena itu, buku ini sangat direkomendasikan sebagai referensi utama bagi siapa pun yang ingin menguasai fondasi formal komputasi dan memahami bagaimana teori abstrak ini membentuk dunia digital yang kita kenal saat ini.