TAMIKA: Jurnal Tugas Akhir Manajemen Informatika & Komputerisasi Akuntansi
Vol. 6 No. 1 (Juni 2026)
https://doi.org/10.46880/tamika.Vol6No1.pp64-70 e-ISSN: 2828-1276

ANALISIS PERBANDINGAN DFA DAN NFA DALAM PENCOCOKAN STRING

Muhammad Rafli Wijaya✉, Zulfahmi Indra, M. Gali Almahdi, Sebastian Saut Marulitua Sinaga
Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Negeri Medan, Medan, Indonesia
Email: rafliwijaya2024@gmail.com

ABSTRACT

*String matching is a fundamental process in pattern recognition systems and large-scale text processing, where computational efficiency significantly affects system performance. This study analyzes the comparison between Deterministic Finite Automata (DFA) and Non-deterministic Finite Automata (NFA) in string matching using the pattern (a|b)*abb. The research was implemented through a web-based simulator developed with JavaScript, HTML, and CSS, and evaluated using eight short test strings representing accepted and rejected inputs, as well as longer strings to observe execution time differences. The results show that DFA and NFA produce identical acceptance outcomes for all test cases, indicating that both automata recognize the same language. However, DFA demonstrates better computational efficiency because each input symbol is processed through a single deterministic transition, whereas NFA requires tracking a set of active states, which increases computational overhead. For longer strings, the execution time gap becomes more pronounced, with DFA remaining consistently faster than NFA. These findings suggest that DFA is more suitable for string matching applications requiring high time efficiency, while NFA offers greater flexibility in transition design.*

Keyword: DFA, NFA, String Matching, Finite State Automata, Computational.

ABSTRAK

*Pencocokan string merupakan proses fundamental dalam sistem pengenalan pola dan pemrosesan teks berskala besar, di mana efisiensi komputasi sangat memengaruhi kinerja sistem. Penelitian ini menganalisis perbandingan Deterministic Finite Automata (DFA) dan Non-deterministic Finite Automata (NFA) dalam pencocokan string dengan pola (a|b)*abb. Penelitian diimplementasikan melalui simulator berbasis web menggunakan JavaScript, HTML, dan CSS, kemudian diuji pada delapan string pendek yang mewakili input diterima dan ditolak, serta string berukuran panjang untuk mengamati perbedaan waktu eksekusi. Hasil pengujian menunjukkan bahwa DFA dan NFA menghasilkan keputusan penerimaan yang identik pada seluruh data uji, sehingga keduanya memiliki kemampuan pengenalan bahasa yang setara. Namun, DFA memperlihatkan efisiensi komputasi yang lebih baik karena setiap simbol input diproses melalui satu transisi deterministik, sedangkan NFA memerlukan pelacakan himpunan state aktif yang menambah beban komputasi. Pada string yang lebih panjang, selisih waktu eksekusi semakin terlihat, dengan DFA tetap lebih cepat dibandingkan NFA. Temuan ini menunjukkan bahwa DFA lebih unggul untuk implementasi pencocokan string yang menuntut efisiensi waktu, sedangkan NFA lebih fleksibel dalam perancangan transisi.*

Kata Kunci: DFA, NFA, Pencocokan String, Finite State Automata, Komputasi.

PENDAHULUAN

Perkembangan teknologi informasi dan komputasi digital saat ini menghasilkan lonjakan data tekstual yang sangat massif. Dalam memproses data berskala besar tersebut, algoritma pencocokan string *(string matching)* menjadi fondasi krusial yang sangat menentukan kecepatan dan keakuratan temu balik informasi (Gil & Santini, 2022).

Secara teoretis, landasan operasional algoritma pencocokan pola sangat bergantung pada pemodelan *Finite State Automata* (FSA), sebuah mesin abstrak komputasi. Dalam praktiknya, dua model utama yang sering dikomparasikan adalah *Deterministic Finite Automata* (DFA) dan *Non-deterministic Finite Automata* (NFA). Kondisi di lapangan sendiri menunjukkan bahwa para pengembang perangkat lunak lebih cenderung mengandalkan model DFA karena keunggulannya pada memberikan satu kepastian transisi untuk setiap simbol masukan, yang menghasilkan eksekusi pencocokan yang sangat cepat, linier, dan terprediksi (Faris Nabhan et al., 2023). Dengan model ini pengimplementasian DFA terbukti sangat efektif dalam merancang alur logika sistem terstruktur, seperti penentuan state pada aplikasi perhitungan uang harian perjalanan dinas maupun sistem berbasis status lainnya (Farhan et al., 2021). Sedangkan model NFA menawarkan fleksibilitas perancangan yang lebih tinggi dibandingkan DFA

karena memungkinkan suatu state memiliki lebih dari satu transisi untuk simbol input yang sama, sehingga lebih efektif dalam merepresentasikan pola dan struktur automata yang kompleks (Kutrib et al., 2022).

Meskipun pemanfaatan DFA dan NFA telah banyak diteliti dan diimplementasikan secara terpisah, kajian terdahulu umumnya hanya berfokus pada fungsionalitas otomata dalam simulasi diagram transisi (Faris Nabhan et al., 2023) atau optimasi minimisasi state berbasis agregasi (Björklund & Cleophas, 2021). Tanpa menganalisis terkait evaluasi komparasi kinerja secara langsung antara NFA dan DFA pada lingkungan pencocokan string yang kompleks. Literatur yang ada belum secara empiris membandingkan efisiensi ketika NFA harus melacak himpunan state aktif yang membebani memori, berhadapan dengan DFA yang mengharuskan pendefinisian transisi untuk seluruh simbol alfabet tanpa pengecualian, sehingga rentan mengalami ledakan status (*state explosion*).

Ketiadaan metrik komparatif mengenai kompleksitas ruang dan waktu antara *Deterministic Finite Automata* (DFA) dan *Non-Deterministic Finite Automata* (NFA) menjadi permasalahan penting dalam pengembangan sistem pencocokan string, karena dapat mengalami inefisiensi, seperti lambatnya iterasi state pada NFA berbasis subset construction maupun tingginya konsumsi memori pada DFA (Farhan et al., 2021). Pemilihan model automata yang kurang tepat terhadap karakteristik pola tertentu dapat berdampak langsung pada penurunan performa sistem secara keseluruhan (Siddique et al., 2022). Oleh sebab itu, penelitian ini dilakukan untuk menganalisis dan membandingkan secara empiris kinerja DFA dan NFA dalam proses pencocokan string berdasarkan kompleksitas perancangan tabel transisi, efisiensi pemrosesan simbol masukan, serta struktur pola yang dibangun (Farhan et al., 2021). Hasil penelitian diharapkan mampu memberikan kontribusi akademis dalam pengayaan literatur terkait optimasi Finite State Automata, sekaligus menjadi rujukan praktis bagi pengembang perangkat lunak dalam menentukan jenis automata yang paling efisien sesuai dengan kebutuhan sistem pengenalan pola yang dikembangkan.

TINJAUAN PUSTAKA

Pencocokan String (*String Matching*)

Algoritma pencocokan pola atau *string matching* merupakan instrumen penting dalam memproses data berskala besar, yang sangat menentukan kecepatan dan keakuratan temu balik informasi. Menurut penelitian oleh (Ikhsan & Fahri Syuhada, 2023), algoritma pencocokan string dapat diterapkan secara efektif bersama *Finite Automata* untuk meningkatkan akurasi sistem, seperti pada aplikasi kamus pencarian kata. Secara teoretis, landasan operasional algoritma pencocokan pola sangat bergantung pada pemodelan *Finite State Automata* (FSA).

Finite State Automata (FSA)

FSA berfungsi sebagai sebuah mesin abstrak komputasi dalam mengeksekusi transisi state. Dalam praktiknya, dua model utama komputasi ini yang sering dikomparasikan adalah *Deterministic Finite Automata* (DFA) dan *Non-deterministic Finite Automata* (NFA). (Wahidin et al., 2021) menggarisbawahi bahwa dengan memadukan algoritma komputasi pencocokan pola ke dalam arsitektur perangkat lunak, sistem dapat memindai teks untuk mencari posisi awal dari karakter dengan kompleksitas komputasi yang efisien.

Deterministic Finite Automata (DFA)

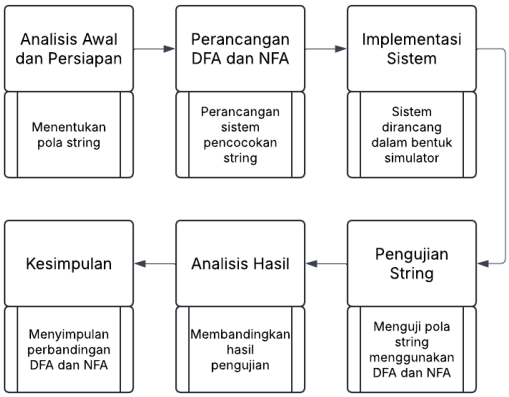
DFA memproses setiap simbol dengan satu operasi pencarian tabel transisi. Hal tersebut mengakibatkan DFA memproses simbol dengan jalur transisi tunggal. Pendekatan jalur tunggal ini menjadikan proses pencocokan cenderung lebih sederhana dan cepat. (Karakchi & Bakos, 2023) mengonfirmasi bahwa mesin DFA sangat unggul untuk komputasi berkinerja tinggi karena determinisme menjamin tidak adanya ambiguitas pemrosesan data.

Non-deterministic Finite Automata (NFA)

Berbeda secara mendasar dari DFA, NFA memerlukan operasi yang lebih kompleks karena harus mengiterasi seluruh *state* dalam himpunan aktif. Selanjutnya, mesin NFA akan mengumpulkan seluruh *state* berikutnya yang mungkin muncul dari input bacaan. Oleh karenanya, kompleksitas per langkah komputasi NFA sangat bergantung pada ukuran himpunan *state* aktif pada proses tersebut (Farhan et al., 2021).

METODE PENELITIAN

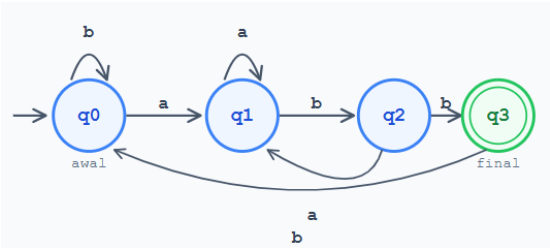
Metodologi penelitian pada penelitian ini disusun sebagai tahapan sistematis yang digunakan untuk membandingkan karakteristik Deterministic Finite Automata (DFA) dan Nondeterministic Finite Automata (NFA) dalam proses pencocokan string. Penelitian dilakukan mulai dari tahap analisis kebutuhan, perancangan automata, implementasi sistem, hingga pengujian dan analisis hasil pengujian. Seluruh tahapan penelitian dirancang agar proses perbandingan DFA dan NFA dapat dilakukan secara terstruktur dan menghasilkan data yang konsisten.



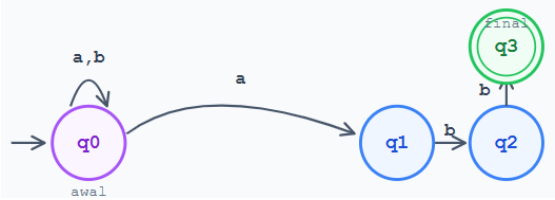
Gambar 1. Alur Penelitian

Analisis Awal dan Persiapan

Penelitian ini dilakukan dengan membandingkan Deterministic Finite Automata (DFA) dan Nondeterministic Finite Automata (NFA) dalam proses pencocokan string. Tahap awal penelitian dimulai dengan menentukan pola string yang akan digunakan sebagai dasar pengujian, yaitu pola (a|b)*abb. Pola tersebut dipilih karena mampu merepresentasikan proses pencocokan string dengan tingkat kompleksitas yang lebih baik dibandingkan pola sederhana, sehingga perbedaan karakteristik DFA dan NFA dapat diamati dengan lebih jelas.



Gambar 2. Diagram DFA

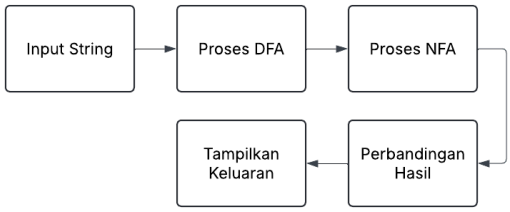


Gambar 3. Diagram NFA

Setelah pola ditentukan, dilakukan penyusunan state, transisi, state awal, dan state akhir pada masing-masing automata. DFA dirancang menggunakan satu jalur transisi untuk setiap simbol input, sedangkan NFA dirancang dengan kemungkinan lebih dari satu transisi pada simbol tertentu. Selain itu, disiapkan pula beberapa data uji berupa string yang sesuai maupun tidak sesuai dengan pola yang digunakan.

Perancangan Sistem

Sistem dirancang dalam bentuk simulator pencocokan string berbasis JavaScript yang terdiri dari modul DFA dan modul NFA. Program menerima input string dari pengguna, kemudian memproses string tersebut menggunakan kedua automata secara terpisah. Hasil pemrosesan ditampilkan dalam bentuk status diterima atau ditolak, state akhir, jejak perpindahan state, dan waktu eksekusi.



Gambar 4. Alur Program

Pada sistem yang dibangun, proses pencocokan string dilakukan dengan membaca input dari kiri ke kanan hingga seluruh simbol selesai diproses. Setelah proses selesai, sistem akan memeriksa apakah state akhir termasuk ke dalam himpunan final state atau tidak. Jika state akhir termasuk final state maka string dinyatakan diterima, sedangkan jika tidak termasuk maka string dinyatakan ditolak.

Pengujian Sistem

Pengujian sistem dilakukan menggunakan beberapa string uji yang dibagi menjadi dua kelompok, yaitu string yang sesuai dengan pola (a|b)*abb dan string yang tidak sesuai dengan pola tersebut. Pengujian dilakukan terhadap DFA dan NFA menggunakan input yang sama agar proses perbandingan dapat dilakukan secara konsisten.

Selain pengujian dasar, dilakukan pula pengujian menggunakan string dengan panjang yang lebih besar untuk melihat proses kerja automata pada jumlah karakter yang lebih banyak. String panjang dibangkitkan secara otomatis menggunakan kombinasi karakter a dan b, kemudian ditambahkan akhiran abb agar sesuai dengan pola yang digunakan.

Analisis Hasil

Analisis data dilakukan dengan membandingkan hasil pengujian DFA dan NFA berdasarkan beberapa aspek, yaitu hasil penerimaan string, jalur transisi state, dan waktu eksekusi program. Data hasil pengujian disajikan dalam bentuk tabel untuk mempermudah proses perbandingan antara kedua automata.

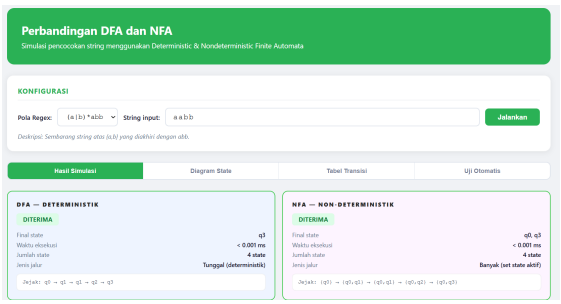
Selain membandingkan waktu eksekusi, analisis juga dilakukan terhadap mekanisme perpindahan state

pada masing-masing automata. DFA dianalisis berdasarkan penggunaan satu jalur transisi pada setiap simbol input, sedangkan NFA dianalisis berdasarkan kemungkinan penggunaan beberapa state aktif secara bersamaan. Analisis tersebut dilakukan untuk melihat perbedaan karakteristik kedua automata dalam proses pencocokan string.

HASIL DAN PEMBAHASAN

Implementasi Sistem

Sistem simulator pencocokan string dibangun dalam bentuk aplikasi berbasis web menggunakan bahasa JavaScript dengan antarmuka HTML dan CSS. Sistem ini terdiri dari dua modul utama, yaitu modul DFA dan modul NFA, yang masing-masing mengimplementasikan automata secara independen tanpa menggunakan pustaka automata pihak ketiga. Seluruh logika transisi, penentuan state aktif, dan keputusan penerimaan string dibangun secara manual menggunakan struktur data objek dan array.



Gambar 5. Halaman Utama Sistem

Pola yang digunakan dalam penelitian ini adalah (a|b)*abb, yaitu sembarang string atas alfabet {a, b} yang diakhiri dengan substring abb. DFA untuk pola ini dirancang memiliki empat state, yaitu q0 sebagai state awal, q1, q2, dan q3 sebagai final state. Tabel fungsi transisi DFA ditunjukkan pada Tabel 1.

Tabel 1. Fungsi Transisi DFA untuk Pola (a|b)*abb

Q	A	B
q0	q1	q0
q1	q1	q2
q2	q1	q3
q3	q1	q0

NFA untuk pola yang sama dirancang dengan empat state, yaitu q0 sebagai state awal, q1, q2, dan q3 sebagai final state. Perbedaan mendasar antara DFA dan NFA terletak pada fungsi transisi: pada NFA, pembacaan simbol 'a' dari state q0 menghasilkan himpunan {q0, q1}, yang berarti terdapat dua kemungkinan state aktif secara bersamaan. Tabel fungsi transisi NFA ditunjukkan pada Tabel 2.

Tabel 2. Fungsi Transisi NFA untuk Pola (a|b)*abb

Q	Input	Stage Berikutnya	Keterangan
q0	a	{q0, q1}	Non-deterministik: dua kemungkinan
q0	b	{q0}	Tetap di q0
q1	b	{q2}	Menuju q2
q2	b	{q3}	Final State (diterima)

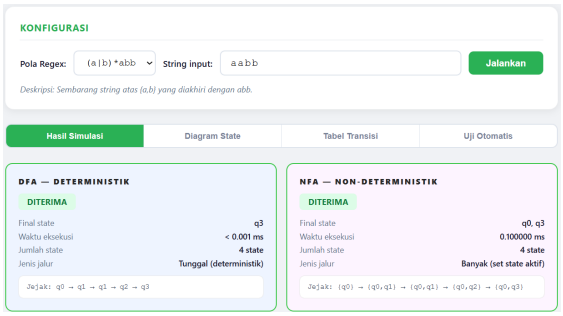
Implementasi DFA pada sistem menggunakan objek berlapis sebagai representasi fungsi transisi δ(q, a) = q', di mana setiap pasangan state dan simbol input dipetakan ke tepat satu state berikutnya. Sementara itu, implementasi NFA menggunakan himpunan (array tanpa duplikasi) untuk menyimpan seluruh kemungkinan state aktif pada setiap langkah pembacaan simbol, sesuai dengan definisi fungsi transisi non-deterministik δ(q, a) ⊆ Q.

Hasil Pengujian String

Pengujian dilakukan terhadap delapan string uji yang dibagi menjadi dua kelompok: empat string yang sesuai dengan pola (a|b)*abb dan empat string yang tidak sesuai. Pengujian dilakukan secara bersamaan pada DFA dan NFA menggunakan input yang identik, sehingga konsistensi hasil antara kedua automata dapat diverifikasi. Hasil pengujian selengkapnya disajikan pada Tabel 3.

Tabel 3. Hasil Pengujian DFA dan NFA terhadap String

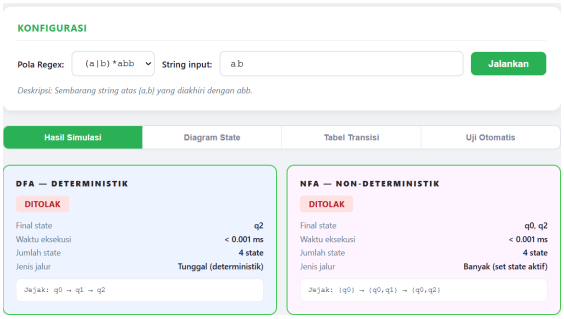
String Uji	Hasil DFA	Hasil NFA
abb	Diterima	Diterima
aabb	Diterima	Diterima
babb	Diterima	Diterima
ababbb	Diterima	Diterima
ab	Ditolak	Ditolak
baa	Ditolak	Ditolak
abba	Ditolak	Ditolak



Gambar 6. Pengujian dengn input aabb

Berdasarkan Tabel 3 dan gambar 6, seluruh string uji menghasilkan keputusan yang konsisten antara DFA dan NFA, baik untuk string yang diterima

maupun yang ditolak. Keempat string yang diterima (abb, aabb, babb, ababbb) seluruhnya memiliki akhiran abb, yang merupakan syarat penerimaan pola (a|b)*abb. Sebaliknya, keempat string yang ditolak tidak memenuhi syarat akhiran tersebut.



Gambar 7. Pengujian dengn input ab

String ab ditolak karena berakhir pada state q2, bukan q3, yang merupakan final state. String abba ditolak meskipun mengandung substring abb, karena setelah abb terdapat karakter tambahan 'a' yang menyebabkan automata berpindah dari final state kembali ke state non-final. Hal ini membuktikan bahwa pola (a|b)*abb mensyaratkan abb berada tepat di akhir string, bukan di tengah.

Tabel 4. Pengujian String Panjang

FSA	String Uji	Status	Waktu
DFA	a*100aabb	Diterima	< 0.0001ms
NFA			0.10000ms
DFA	b*1000abb	Diterima	0.10000ms
NFA			0.40000ms
DFA	A*10000abb	Diterima	0.90000ms
NFA			2.30000ms

Selain pengujian terhadap string pendek, dilakukan pula pengujian menggunakan string dengan panjang karakter yang jauh lebih besar untuk mengamati perbedaan waktu eksekusi DFA dan NFA secara lebih nyata. String panjang dibangkitkan secara otomatis dengan menggabungkan pengulangan karakter tunggal sebanyak ratusan hingga ribuan kali, kemudian ditambahkan akhiran *abb* agar string memenuhi pola *(a|b)*abb* dan dinyatakan diterima oleh kedua automata.

Hasil pengujian menunjukkan bahwa pada string *a* yang diulang sebanyak 100 kali diikuti akhiran *abb* (total 103 karakter), DFA menyelesaikan proses pencocokan dengan waktu kurang dari 0,0001 ms, sedangkan NFA memerlukan waktu 0,1000 ms. Perbedaan yang cukup signifikan tampak pada string *b* yang diulang sebanyak 1.000 kali diikuti akhiran *abb* (total 1.003 karakter), DFA membutuhkan waktu 0,1000 ms, sedangkan NFA membutuhkan waktu 0,4000 ms atau empat kali lebih lama dibandingkan DFA. Kemudian pada pengujian string *a* yang diulang sebanyak 1000 kali (total 10003 karakter), DFA selesai degan waktu 0.9000ms, sedangka NFA selesai dengan waktu 2.3000ms. Kedua string tersebut diterima oleh DFA maupun NFA, membuktikan konsistensi hasil penerimaan tidak terpengaruh oleh panjang string.

Pola perbedaan waktu ini mengonfirmasi bahwa beban tambahan pada NFA berasal dari operasi pengelolaan himpunan state aktif di setiap langkah pembacaan simbol. Semakin panjang string yang diproses, semakin besar akumulasi selisih waktu antara DFA yang hanya melakukan satu pencarian transisi per simbol dengan NFA yang harus mengiterasi seluruh elemen himpunan state aktif sebelum melanjutkan ke simbol berikutnya.

Perbandingan DFA dan NFA

Perbandingan antara DFA dan NFA dilakukan berdasarkan tiga aspek utama: hasil penerimaan string, jejak perpindahan state, dan karakteristik komputasi.

Tabel 5. Perbandingan Jejak State DFA dan NFA

String	Jejak DFA	Jejak NFA
Abb	q0→q1→q2→q3	{q0}→{q0,q1}→{q0,q2}→{q0,q3}
Aabb	q0→q1→q1→q2→q3	{q0}→{q0,q1}→{q0,q1}→{q0,q2}→{q0,q3}
Babb	q0→q0→q1→q2→q3	{q0}→{q0}→{q0,q1}→{q0,q2}→{q0,q3}
Ab	q0→q1→q2	{q0}→{q0,q1}→{q0,q2}
abba	q0→q1→q2→q3→q1	{q0}→{q0,q1}→{q0,q2}→{q0,q3}→{q0,q1}

Dari Tabel 5 terlihat perbedaan mendasar pada representasi jejak state. DFA selalu menunjukkan satu state aktif pada setiap langkah, sedangkan NFA menunjukkan himpunan state yang dapat berisi lebih dari satu elemen. Sebagai contoh, pada pembacaan string aabb, DFA melewati jalur tunggal q0 → q1 → q1 → q2 → q3, sementara NFA memproses himpunan {q0} → {q0, q1} → {q0, q1} → {q0, q2} → {q0, q3}. Keduanya menghasilkan keputusan yang sama, tetapi mekanisme internalnya berbeda secara fundamental.

Perbandingan karakteristik umum antara DFA dan NFA untuk pola (a|b)*abb disajikan pada Tabel 6.

Tabel 6. Perbandingan Karakteristik DFA dan NFA

Aspek	**DFA**	**NFA**
Status penerimaan string	Identik untuk semua string uji	Identik untuk semua string uji
Jumlah state	4 state (q0, q1, q2, q3)	4 state (q0, q1, q2, q3)
Jumlah state aktif per langkah	1 state aktif	Banyak state aktif (set)
Dead state eksplisit	Tidak ada (kembali ke q0/q1)	Tidak ada (set kosong = tolak)
Kompleksitas jalur	Tunggal dan deterministik	Bercabang dan non-deterministik
Representasi transisi	Satu fungsi δ per simbol per state	Fungsi δ menghasilkan himpunan state
Kecepatan eksekusi	Lebih cepat (satu jalur)	Sedikit lebih lambat (tracking set)
Kemudahan perancangan	Lebih kompleks (perlu eliminasi NDA)	Lebih ringkas dan alami

Pembahasan

Hasil pengujian pada Tabel 3 menunjukkan bahwa DFA dan NFA menghasilkan keputusan penerimaan string yang identik untuk seluruh string uji yang digunakan. Hal ini konsisten dengan teorema ekuivalensi antara DFA dan NFA yang menyatakan bahwa untuk setiap NFA terdapat DFA yang mengenali bahasa yang sama. Dengan demikian, perbedaan antara DFA dan NFA bukan terletak pada kemampuan pengenalan bahasa, melainkan pada mekanisme internal pemrosesan string.

Perbedaan pertama yang paling terlihat adalah cara kedua automata merepresentasikan state aktif. DFA selalu berada pada tepat satu state di setiap langkah, sehingga jejak transisinya bersifat linear dan mudah ditelusuri. NFA, sebaliknya, mempertahankan himpunan state aktif yang dapat tumbuh atau menyusut bergantung pada simbol yang dibaca. Pada pola (a|b)*abb, NFA mempertahankan state q0 dalam himpunan aktif sepanjang proses berlangsung, mencerminkan sifat prefix bebas dari pola tersebut, yakni bahwa string valid dapat diawali oleh sembarang kombinasi a dan b.

Perbedaan kedua berkaitan dengan kemudahan perancangan. NFA untuk pola (a|b)*abb memiliki empat aturan transisi yang intuitif dan langsung merepresentasikan struktur pola. DFA untuk pola yang sama juga memiliki empat state, namun setiap state harus mendefinisikan transisi untuk seluruh simbol alfabet tanpa pengecualian, sehingga proses perancangannya memerlukan analisis yang lebih teliti terhadap seluruh kemungkinan urutan simbol yang dapat membawa automata menuju atau meninggalkan final state.

Perbedaan ketiga berkaitan dengan efisiensi eksekusi. DFA memproses setiap simbol dengan satu operasi pencarian tabel transisi, sehingga DFA memproses simbol dengan jalur transisi tunggal sehingga proses pencocokan cenderung lebih sederhana dan cepat. NFA memerlukan operasi yang lebih kompleks karena harus mengiterasi seluruh state dalam himpunan aktif dan mengumpulkan seluruh state berikutnya yang mungkin, sehingga kompleksitas per langkah bergantung pada ukuran himpunan state aktif. Pada implementasi berbasis subset construction seperti yang digunakan dalam penelitian ini, NFA dieksekusi dengan melacak himpunan state secara eksplisit, yang menjadikan kompleksitas totalnya setara dengan konstruksi DFA dari NFA tersebut.

KESIMPULAN

Penelitian menyimpulkan bahwa meskipun DFA dan NFA memiliki tingkat akurasi yang setara dalam pencocokan string, keduanya memiliki karakteristik operasional yang berbeda DFA unggul dalam kecepatan eksekusi karena sifat deterministiknya yang linier, sedangkan NFA menawarkan fleksibilitas perancangan yang lebih baik meski memerlukan beban komputasi tambahan saat melacak state aktif. Pemilihan model automata harus disesuaikan dengan kebutuhan sistem, di mana DFA lebih efektif untuk aplikasi yang mengutamakan performa real-time, sementara NFA lebih relevan untuk kebutuhan desain pola yang kompleks.

DAFTAR PUSTAKA

Björklund, J., & Cleophas, L. (2021). Aggregation-based minimization of finite state automata. *Acta Informatica*, *58*(3), 177–194. https://doi.org/10.1007/s00236-019-00363-5

Farhan, S., Fitri, I., Infoman, F., Studi Informatika, P., Sawo Manila Pasar Minggu, J., & Selatan, J. (2021). Implementasi Transisi Finite State Automata dengan Aplikasi Mesin Abstrak DFA dan NFA Berbasis Android. *Jurnal Ilmu-Ilmu Informatika Dan Manajemen STMIK*.

Faris Nabhan, M., Avrilia Lantana, D., Zidni, P., Arofi, A., Habibbullah, P., Frazza Mukti, A., & Iqbal Khadavi, M. (2023). Simulator Mesin Deterministic Finite Automata (DFA)

Berdasarkan Diagram Transisi Menggunakan Python. *Jurnal Sistem Komputer Dan Kecerdasan Buatan*.

Gil, J. A., & Santini, S. (2022). Matching Regular Expressions on uncertain data. *Algorithmica*, *84*(2), 532–564. https://doi.org/10.1007/s00453-021-00906-8

Ikhsan, M., & Fahri Syuhada, M. (2023). Implementation of String Matching Algorithm with Finite Automata in The Indonesian-Korean Dictionary Application Article History. *Sainstek: Jurnal Sains dan Teknologi*, *15*(2), 2580–278.

Karakchi, R., & Bakos, J. D. (2023). NAPOLY: A Non-deterministic Automata Processor OverLaY. *ACM Transactions on Reconfigurable Technology and Systems*, *16*(3). https://doi.org/10.1145/3593586

Kutrib, M., Malcher, A., & Schneider, C. (2022). Finite automata with undirected state graphs. *Acta Informatica*, *59*(1), 163–181. https://doi.org/10.1007/s00236-021-00402-0

Siddique, F. A., Tracy, T. J., Brunelle, N., & Skadron, K. (2022). Deterministic vs. Non Deterministic Finite Automata in Automata Processing. *ArXiv*. http://arxiv.org/abs/2210.10077

Wahidin, W., Yasin, V., & Haroen, R. (2021). Perancangan sistem informasi pengelolaan lapangan futsal berbasis web dengan metode rapid application development menggunakan algoritma string matching di maestro futsal kemayoran jakarta. *Journal of Information System, Informatics and Computing*, *5*(1), 1. https://doi.org/10.52362/jisicom.v5i1.375