

## Pengantar ke NFA dan DFA

Sebelum kita membahas diagram secara detail, mari kita ingatkan apa itu NFA (Non-deterministic Finite Automaton) dan DFA (Deterministic Finite Automaton). Ini adalah konsep dasar dalam teori komputasi, khususnya untuk mengenali bahasa regular.

- **NFA:** Mesin otomatis yang non-deterministik, artinya dari satu state (keadaan), untuk input yang sama, bisa ada beberapa kemungkinan transisi (atau bahkan transisi  $\epsilon$ , yaitu tanpa input). NFA lebih mudah dirancang untuk pola kompleks karena fleksibilitasnya, tapi lebih sulit diimplementasikan secara langsung karena non-determinisme.
- **DFA:** Versi deterministik dari NFA, di mana dari setiap state, untuk setiap input, hanya ada tepat satu transisi. DFA ekuivalen dengan NFA (setiap NFA bisa dikonversi ke DFA), tapi DFA biasanya memiliki lebih banyak state. DFA lebih efisien untuk simulasi karena tidak ada cabang non-deterministik.

Dalam konteks permintaan Anda (website file converter DOCX to PDF), saya menggunakan contoh sederhana: mesin yang mengenali string persis "docx" atau "pdf". Ini bisa digunakan untuk validasi ekstensi file (misalnya, memeriksa apakah input adalah ".docx" dan output ".pdf"). Bahasa regular yang dikenali adalah  $\{ \text{"docx"}, \text{"pdf"} \}$ , dengan alfabet  $\Sigma = \{c, d, f, o, p, x\}$  (huruf-huruf yang muncul).

Saya membangun NFA menggunakan konstruksi Thompson (untuk regular expression "docx|pdf"), lalu mengonversinya ke DFA menggunakan subset construction (konstruksi himpunan bagian). Penjelasan ini akan lengkap, termasuk langkah-langkah konstruksi, state, transisi, dan representasi diagram.

## Penjelasan Lengkap NFA

NFA dirancang dengan dua cabang paralel (untuk "docx" dan "pdf"), dihubungkan melalui transisi  $\epsilon$  dari state start.

### Komponen NFA:

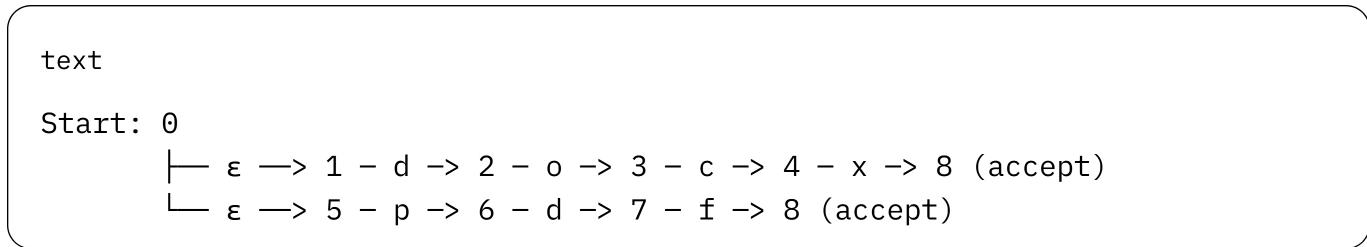
- **States (Q):** {0, 1, 2, 3, 4, 5, 6, 7, 8}
    - 0: State start (initial).
    - 1-4: Path untuk "docx".
    - 5-7: Path untuk "pdf".
    - 8: State accept (final), satu-satunya state akhir untuk kedua path.
  - **Alfabet ( $\Sigma$ ):** {c, d, f, o, p, x}.
  - **State Start (q0):** 0.
  - **States Accept (F):** {8}.
  - **Fungsi Transisi ( $\delta$ ):**
    - $\delta(0, \varepsilon) = \{1, 5\}$  (cabang non-deterministik ke awal "docx" atau "pdf").
    - $\delta(1, d) = \{2\}$
    - $\delta(2, o) = \{3\}$
    - $\delta(3, c) = \{4\}$
    - $\delta(4, x) = \{8\}$
    - $\delta(5, p) = \{6\}$
    - $\delta(6, d) = \{7\}$
    - $\delta(7, f) = \{8\}$
- Tidak ada transisi lain; input invalid menyebabkan reject.

**Cara Kerja NFA:**

- Mulai di state 0.
- Dari 0, tanpa input ( $\epsilon$ ), bisa pindah ke 1 (untuk "docx") atau 5 (untuk "pdf").
- Untuk input "docx":  $0 \xrightarrow{(\epsilon)} 1 \xrightarrow{(d)} 2 \xrightarrow{(o)} 3 \xrightarrow{(c)} 4 \xrightarrow{(x)} 8$  (accept).
- Untuk input "pdf":  $0 \xrightarrow{(\epsilon)} 5 \xrightarrow{(p)} 6 \xrightarrow{(d)} 7 \xrightarrow{(f)} 8$  (accept).
- Input lain (misalnya "doc") akan stuck sebelum mencapai 8, sehingga reject.
- Non-determinisme: Mesin "mencoba" semua path secara paralel; jika setidaknya satu path mencapai accept, string diterima.

#### **Representasi Diagram NFA (Teks ASCII):**

Diagram ini menggambarkan graf secara linear. Panah menunjukkan transisi.



- Ini seperti pohon dengan dua cabang dari 0.

#### **Kode Graphviz (DOT) untuk Visualisasi NFA:**

Gunakan ini di tool seperti Graphviz Online untuk gambar grafis.

```

text

digraph NFA {
    rankdir=LR;
    node [shape=doublecircle]; 8
    node [shape=circle];
    invisible [style=invis, shape=none];
    invisible -> 0 [label="start"];
    0 -> 1 [label="ε"];
    0 -> 5 [label="ε"];
    1 -> 2 [label="d"];
    2 -> 3 [label="o"];
    3 -> 4 [label="c"];
    4 -> 8 [label="x"];
    5 -> 6 [label="p"];
    6 -> 7 [label="d"];
    7 -> 8 [label="f"];
}

```

- State 8 adalah lingkaran ganda (accept).
- Transisi  $\epsilon$  ditunjukkan tanpa input.

## Proses Konversi NFA ke DFA (Subset Construction)

Untuk mengonversi NFA ke DFA, kita gunakan algoritma subset construction:

1. **Hitung  $\epsilon$ -closure:** Himpunan state yang bisa dicapai dari state tertentu hanya dengan transisi  $\epsilon$  (termasuk dirinya sendiri).
  - $\epsilon$ -closure(0) = {0, 1, 5} (karena dari 0 ke 1 dan 5 via  $\epsilon$ ).
  - $\epsilon$ -closure(1) = {1},  $\epsilon$ -closure(2) = {2}, dst. (tidak ada  $\epsilon$  lain).
2. **State DFA awal:**  $\epsilon$ -closure(state start NFA) = {0, 1, 5}. Sebut ini state A (atau 0 di diagram saya).
3. **Untuk setiap state DFA (himpunan state NFA) dan setiap simbol a di  $\Sigma$ :**
  - Hitung  $\text{move}(S, a)$  = gabungan  $\delta(q, a)$  untuk q di S.
  - Kemudian  $\epsilon$ -closure( $\text{move}(S, a)$ ) menjadi state DFA baru.
4. Ulangi sampai tidak ada state baru. State DFA yang mengandung state accept NFA menjadi accept.
5. Jika ada input tanpa transisi, itu menuju trap state (reject), tapi di diagram saya, saya buat partial (hanya transisi valid).

**Langkah-langkah Konversi Detil:**

- State DFA 0: {0,1,5}
  - move({0,1,5}, d) = {2} (dari 1),  $\epsilon$ -closure = {2}  $\rightarrow$  State 1: {2}
  - move({0,1,5}, p) = {6} (dari 5),  $\epsilon$ -closure = {6}  $\rightarrow$  State 2: {6}
  - (Tidak ada transisi untuk c, f, o, x dari sini).
- State DFA 1: {2}
  - move({2}, o) = {3},  $\epsilon$ -closure = {3}  $\rightarrow$  State 3: {3}
- State DFA 2: {6}
  - move({6}, d) = {7},  $\epsilon$ -closure = {7}  $\rightarrow$  State 4: {7}
- State DFA 3: {3}
  - move({3}, c) = {4},  $\epsilon$ -closure = {4}  $\rightarrow$  State 5: {4}
- State DFA 4: {7}
  - move({7}, f) = {8},  $\epsilon$ -closure = {8}  $\rightarrow$  State 6: {8}
- State DFA 5: {4}
  - move({4}, x) = {8},  $\epsilon$ -closure = {8}  $\rightarrow$  State 6: {8}
- State 6: {8} (accept, karena mengandung 8). Tidak ada transisi keluar lebih lanjut.

DFA ini minimal (tidak ada state redundan), dengan 7 state.

## Penjelasan Lengkap DFA

DFA adalah versi deterministik, tanpa  $\epsilon$ , dan setiap input punya tepat satu path.

**Komponen DFA:**

- **States (Q)**: {0, 1, 2, 3, 4, 5, 6} (masing-masing mewakili himpunan state NFA di atas).
  - 0: {0,1,5} (start).
  - 1: {2}
  - 2: {6}
  - 3: {3}
  - 4: {7}
  - 5: {4}
  - 6: {8} (accept).
- **Alfabet ( $\Sigma$ )**: Sama seperti NFA.
- **State Start ( $q_0$ )**: 0.
- **States Accept (F)**: {6}.
- **Fungsi Transisi ( $\delta$ )**:
  - $\delta(0, d) = 1$
  - $\delta(0, p) = 2$
  - $\delta(1, o) = 3$
  - $\delta(2, d) = 4$
  - $\delta(3, c) = 5$
  - $\delta(4, f) = 6$
  - $\delta(5, x) = 6$

Transisi lain (invalid) menuju trap state implisit (reject).

**Cara Kerja DFA:**

- Mulai di 0.
- Untuk "docx":  $0 \rightarrow (d) 1 \rightarrow (o) 3 \rightarrow (c) 5 \rightarrow (x) 6$  (accept).
- Untuk "pdf":  $0 \rightarrow (p) 2 \rightarrow (d) 4 \rightarrow (f) 6$  (accept).
- Input invalid (misalnya dari 0 dengan 'c')  $\rightarrow$  trap  $\rightarrow$  reject.
- Deterministik: Selalu satu path unik untuk setiap input.

#### Representasi Diagram DFA (Teks ASCII):

```
text

Start: 0 - d -> 1 - o -> 3 - c -> 5 - x -> 6 (accept)
      |
      └ p -> 2 - d -> 4 - f -> 6 (accept)
```

- Dua cabang deterministik dari 0.

#### Kode Graphviz (DOT) untuk Visualisasi DFA:

```
text

digraph DFA {
    rankdir=LR;
    node [shape=doublecircle]; 6
    node [shape=circle];
    invisible [style=invis, shape=none];
    invisible -> 0 [label="start"];
    0 -> 1 [label="d"];
    0 -> 2 [label="p"];
    1 -> 3 [label="o"];
    3 -> 5 [label="c"];
    5 -> 6 [label="x"];
    2 -> 4 [label="d"];
    4 -> 6 [label="f"];
}
```

- State 6 adalah lingkaran ganda.

#### Aplikasi di Website File Converter

- NFA/DFA ini bisa diimplementasikan di backend (misalnya JavaScript atau Python) untuk validasi: Cek jika ekstensi input = "docx" (accept lalu convert), output = "pdf".
- Keuntungan DFA: Lebih cepat dijalankan karena deterministik, cocok untuk web app.
- Jika ingin lebih kompleks (misalnya handle full filename seperti "file.docx"), kita bisa extend regular expression, tapi prinsip sama.

Jika Anda ingin contoh kode implementasi (misalnya Python simulator untuk NFA/DFA), atau modifikasi diagram (misalnya tambah trap state eksplisit), beri tahu! Untuk visual grafis, salin DOT code ke Graphviz tool. Jika Anda ingin saya bantu generate image dari diagram ini, konfirmasikan ya.