第 9 講 非決定的有限状態機械

林　恒俊

非決定的と決定的

- 非決定的機械は決して特別なものではない。決定的機械の定義を素直に拡張すると自然に非決定的機械に辿り着く。
- また任意の言語を受理するような機械を設計しようとすると非決定的機械なら容易に構成することが可能である。
- 正規言語や正規表現にしても言語を定義する操作を適用する順序はあらかじめ決まっている訳ではない。様々な適用が可能であるから様々な語を生成できる。
- これらの点からいっても非決定的機械の方がより一般的であり、決定的機械の方が特別の場合であることが理解できよう。

非決定的有限状態機械

- NFA M は形式的に $M = (Q, \Sigma, \Delta, q_0, F)$ の 5 つ組で定義される。ここで
  - Q は状態の有限集合
  - Σ はアルファベットあるいは終端記号集合
  - Δ は状態遷移を代表する関係で

$$\Delta = \{(q, \sigma, q') \mid q, q' \in Q, \sigma \in \Sigma^*\}$$

  - q_0 は初期状態、ただし $q_0 \in Q$
  - F は終了状態集合、ただし $F \subseteq Q$

- 状態遷移関係 Δ で、q は現状態、q' は次状態、σ は入力記号列を示している。
- この定義は DFA の定義と状態遷移関数の部分を除いて全くかわらない。NFA と DFA の違いは状態遷移の動作のみということになる。その詳細は

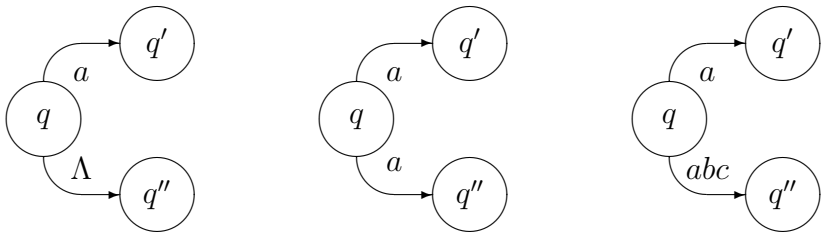
種類	状態遷移	入力	次状態
DFA	関数	現状態、1 記号	1 状態
NFA	関係	現状態、記号列 (空列含む)	複数の状態可

である。

非決定的な動作

- NFA の動作は基本的には DFA と同じである。現状態と入力記号列で次状態を決定し状態遷移する。しかし次状態が複数個ある場合に非決定的な動作をするところが異なっている。
- DFA では状態遷移毎に入力テープが必ず 1 つ進む。NFA の状態遷移の入力が記号列になっているため常に 1 つだけ進むとは限らない。
  - 入力テープは記号列の長さ分一気に進む。
  - 記号列が空列の場合はテープは進まない。

  最後の場合を特に**空遷移** (**Λ-transition**) という。空遷移では入力が空記号 Λ のため遷移関数のパラメータとして現状態が重要である。
- 次のような NFA は入力記号が a なら q' と q'' のいずれにも遷移する可能性がある。取りうる次状態が複数なら非決定的に動作する。



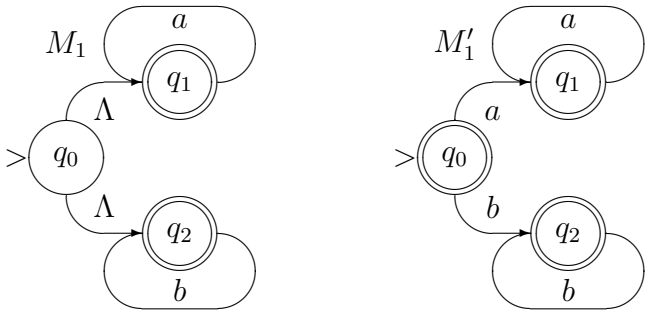
非決定的動作では可能なすべての状態遷移を試みる。試みたすべての動作状態中に

- 1 個でも受理状態があれば全体で入力記号列を受理
- すべての動作状態中に受理状態がなければ拒否

と判定する。

NFA の実例—その 1

- 次の NFA $M_1 = (\{q_0, q_1, q_2\}, \{a, b\}, \Delta, q_0, \{q_1, q_2\})$ を考える。ただし $\Delta = \{(q_0, \Lambda, q_1), (q_0, \Lambda, q_2), (q_1, a, q_1), (q_2, b, q_2)\}$ である。M_1 の状態遷移図は次のように表示される。



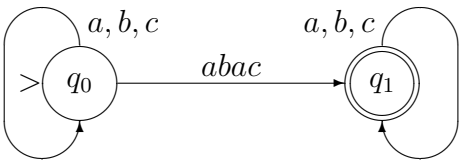
- この遷移図から理解されるように M_1 の受理する言語は $\{a\}^* \cup \{b\}^*$ であり正規表現では `a*|b*` となる。
- 入力とは無関係に、いきなり初期状態 q_0 から空遷移で q_1 と q_2 のいずれかに移行するするように構成されているため非決定的な動作を行わなければならない。一度いずれかの状態に移行してしまえば後は決定的動作を行う。ここでは非決定的動作が集合和演算あるいは選択演算を実現している。言語の構成と NFA の構成が素直に結びついていることが理解できよう。
- 同じ言語を受理する DFA M_1' は $M_1' = (\{q_0, q_1, q_2\},\ \{a, b\},\ \delta,\ q_0,\ \{q_0, q_1, q_2\})$ ただし $\delta = \{(q_0, a, q_1),\ (q_0, b, q_2),\ (q_1, a, q_1),\ (q_2, b, q_2)\}$ である。
- 一般に受理する言語に空列を含む場合空列についても正常に受理するために初期状態が終了状態に含められる。

NFA の実例—その 2

- NFA M_2 の定義は

$$M_2 = (\{q_0, q_1\}, \{a, b, c\}, \Delta, q_0, \{q_1\})$$
$$\Delta = \{(q_0, a, q_0), (q_0, b, q_0), (q_0, c, q_0), (q_0, abac, q_1),$$
$$(q_1, a, q_1), (q_1, b, q_1), (q_1, c, q_1)\}$$

  である。M_2 の状態遷移図は次に示される。



  この状態遷移図から M_2 が受理する言語は $\{a, b, c\}^* abac\{a, b, c\}^*$ であることが判断できる。この言語は $abac$ を部分列として含む記号列を要素とする。

- M_2 は q_0 で a を入力したときに、次状態が q_0 と q_1 のいずれにでも推移する可能性があるため非決定的動作が必要である。

- M_2 の受理する言語は**有限状態機械の設計**で検討した機械 M_{abac} が受理するものと同一である。言語を与えられた時それを受理する FSA は NFA なら素直に構成可能であることがこの例からも理解できる。

NFA の動作の形式的定義

- DFA と同様に NFA の動作についても形式的定義を行うことができる。ただし状態遷移するときの入力が記号列であるため少し注意が必要である。以下では機械の動作状態やテープ構成について DFA と同じ記法を使用する。

- ある NFA M について $\mathcal{C}_1 = (q_1, [p_1, x]), \mathcal{C}_2 = (q_2, [p_2, x])$ ただし $\mathcal{C}_1, \mathcal{C}_2 \in C(M)$ を M の任意の 2 個の動作状態とし $\sigma \in \Sigma^*$ をアルファベット Σ 上の記号列とすると

$$\begin{cases} p_1 + |\sigma| = p_2 \quad \text{かつ} \quad x(p_1 + i - 1) = \sigma(i) \quad (1 \le i \le |\sigma|) & (1) \\ (q_1, \sigma, q_2) \in \Delta & (2) \end{cases}$$

の時 $\mathcal{C}_1 \vdash_M \mathcal{C}_2$ と表示し動作状態 $\mathcal{C}_1$ から動作状態 $\mathcal{C}_2$ に進むという。

上記 (1) は最初にテープのヘッドの位置以降の記号列が状態遷移表にある記号列と一致していることを示している。そしてヘッドがこの記号列の後の位置に移動するまでテープが前進する。なお遷移表の記号が空列の場合はテープは前進せず入力記号に無関係にステップが進む。

- $\vdash_M$ は機械動作状態集合間で定義される関係演算で $\vdash_M \subseteq C(M) \times C(M)$ である。さらに一連の機械動作状態 $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_n \in C(M)$ に

$$\mathcal{C}_1 \vdash_M \mathcal{C}_2, \ldots, \mathcal{C}_{n-1} \vdash_M \mathcal{C}_n$$

という一連の関係が同時に存在すれば

$$\mathcal{C}_1 \vdash_M \mathcal{C}_2 \vdash_M \cdots \vdash_M \mathcal{C}_n = \mathcal{C}_1 \vdash_M^* \mathcal{C}_n$$

によりこの機械の計算過程を示す。

- NFA は初期構成 $\mathcal{C}_0 = (q_0, [1, x])$ から動作を開始する。
- NFA は動作状態 $\mathcal{C}_h = (q, [p, x]) \in C(M)$ で次状態が定義されていない時停止する。すなわちヘッドがテープの最後に到達した $p = |x|+1$ の場合や $\Delta(q, x(p))$ が未定義の場合である。この時 NFA は停止動作状態 $\mathcal{C}_h$ に達したという。

非決定的動作

- DFA と異なり NFA の場合にはある機械動作状態 $\mathcal{C}_1$ について次動作状態が複数個 $\mathcal{C}_{21}, \mathcal{C}_{22}, \ldots$ 存在してもよい。

$$\mathcal{C}_1 \vdash_M \mathcal{C}_{21}, \mathcal{C}_1 \vdash_M \mathcal{C}_{22}, \ldots \quad \text{ただし } C_{21} \neq C_{22}, \ldots$$

- DFA の初期動作状態から $\vdash_M$ 演算による一連の動作状態式の代わりに、NFA では動作状態の列が枝分かれしていくつかの列が並列することになる。例えば $\mathcal{C}_{i-1} \vdash_M \mathcal{C}_i, \mathcal{C}_i \vdash_M \mathcal{C}_{j1}, \mathcal{C}_i \vdash_M \mathcal{C}_{j2}$ が同時に成立する場合

$$\begin{array}{c} \cdots \vdash_M \mathcal{C}_{i-1} \vdash_M \mathcal{C}_i \vdash_M \mathcal{C}_{j1} \vdash_M \cdots \\ \downarrow \text{分岐} \\ \mathcal{C}_i \vdash_M \mathcal{C}_{j2} \vdash_M \cdots \end{array}$$

- ある終了状態 $f \in F$ について

$$(q_0, [1, x]) \vdash_M^* (f, [|x| + 1, x])$$

という計算過程がいずれかの枝中に存在すればNFA M は記号列 x を受理するという。

- ある有限状態機械が定義する言語はその機械が受理する記号列の集合である。この定義はDFAのそれと同じである。

- なお受理する場合と拒否する場合の定義が対称ではないため受理は容易に検証できるが拒否の検証はより難しい。受理する計算過程が1つでもあれば受理を結論づけられるのに対して拒否を結論づけるためにはすべての計算過程について拒否することを検証しなければならない。

NFA の動作例

- 上記の M_2 について $abaca$ という記号列を入力したときの計算過程を次に示す。初期動作状態は $(q_0, [1, abaca])$ である。

- 計算過程は2つに枝分かれする。最初の計算過程は次のように記号列 $abac$ で状態遷移するものである。この場合入力記号列は受理される。

$$\begin{aligned}(q_0, [1, abaca]) &\vdash_{M_2} (q_1, [5, abaca]) \\ &\vdash_{M_2} (q_1, [6, abaca])\end{aligned}$$

- もう1つの計算過程は次のように最初の状態を繰返すものである。

$$\begin{aligned}(q_0, [1, abaca]) &\vdash_{M_2} (q_0, [2, abaca]) \\ &\vdash_{M_2} (q_0, [3, abaca]) \\ &\vdash_{M_2} (q_0, [4, abaca]) \\ &\vdash_{M_2} (q_0, [5, abaca]) \\ &\vdash_{M_2} (q_0, [6, abaca])\end{aligned}$$

この場合入力記号列は最後まで処理されるが q_0 状態は終了状態集合には含まれないので記号列 $abaca$ は受理されない。

- 結論として $abaca$ を受理する計算過程が存在するので M_2 は $abaca$ を受理する。

考察

FSA を記号を判別する機械**受理器** (**acceptor**) として考えれば DFA と NFA の差は大きい。記号列を受理する過程を計算する方法が異なっているからである。しかし FSA を記号列を生成する**生成器** (**generator**) と見なせば DFA と NFA の違いはほとんど無視できる。

生成器としての FSA は次のようなものである。FSA を構文図の変形したものと捉え初期状態から着目点が枝を辿って次状態に移動するものと考える。その時枝につけられた記号が記号列として出力される。最終的に次状態に移行できなくなるかいずれかの終了状態に到達すれば停止してもよい。終了状態で停止した場合出力された記号列がその FSA が定義する言語の要素である。ある状態から複数の枝が出ている場合には適当な枝を選べばよい。枝につけられた記号は選び方とは無関係で複数の枝に同じ記号がつけられているかあるいはいずれかの枝に空記号がつけられていれば NFA ということになる。