『論理と形式化』 資料その 4
亀山幸義

1 Prolog の実行: 導出原理

Prolog は Horn 節の集合に対する自動証明システムであり、それをプログラム言語と見なしたものである。

この自動証明手続きは、導出 (resolution) と呼ばれる。Prolog の場合は，特に SLD 導出と呼ばれる。)。導出の簡単な場合について、この資料で説明する。

2 導出原理 (命題論理の範囲)

Horn 節が命題論理の範囲内であるとする。つまり、述語記号 $P, Q, ...$ がすべて「アリティ0 (引数が 0 個)」の場合である。0 引数の述語は、公式には $P()$ や $Q()$ と書くことにしていたが、括弧を省略して、$P, Q, ...$ と書くことにする。

2.1 例 1

以下の Prolog プログラムを考える。(1 つ 1 つの Horn 節には H1, H2, ... という名前をつけた。)

```
H1   P :- Q.
H2   Q :- R.
H3   R.
```

これらを、論理式として表現すると、以下の通りである。

$$\begin{array}{ll} H1 & Q \supset P \\ H2 & R \supset Q \\ H3 & R \end{array}$$

この例では、項に対する変数 $x, y, \cdots$ が含まれないので、$\forall$ は現れない。

H1, H2, H3 をつかって論理式 P を導出すると以下のようになる。

$$\dfrac{\dfrac{\overset{(H3)}{R} \quad \overset{(H2)}{R \supset Q}}{Q} \supset\text{-E} \quad \overset{(H1)}{Q \supset P}}{P} \supset\text{-E}$$

これを Prolog でやってみよう。

ゴール`?- P.`の実行 (導出):

ゴールは、「コンマで区切られた原子論理式たち」であり、それをここでは集合 G_i と書くことにする。

0 最初: $G_0 = \{P\}$.

1 P に対して (H1) を使って: $G_1 = \{Q\}$.

2 Q に対して (H2) を使って: $G_2 = \{R\}$.

3 R に対して (H3) を使って: $G_3 = \{\}$.

4 ゴールが空集合になったので、実行は成功して終わる。

証明図を下からつくっていることがわかる。

ゴール`?- P,Q.` の実行 (導出): これは、論理式で書けば、$P \wedge Q$ であるかどうかを導出するものである。

0 最初: $G_0 = \{P, Q\}$.
1 P に対して (H1) を使って: $G_1 = \{Q, Q\}$.
2 Q に対して (H2) を使って: $G_2 = \{R, Q\}$.
3 R に対して (H3) を使って: $G_3 = \{Q\}$.
4 Q に対して (H2) を使って: $G_4 = \{R\}$.
5 R に対して (H3) を使って: $G_5 = \{\}$.
6 ゴールが空集合になったので、実行は成功して終わる。

2.2 例 2

以下の Prolog プログラムを考える。

```
H1   Q :- R, S.
H2   R :- S.
H3   S.
```

これらを、論理式として表現すると、以下の通りである。

$$\begin{array}{ll} H1 & (R \wedge S) \supset Q \\ H2 & S \supset R \\ H3 & S \end{array}$$

ゴール`?- Q.` の実行 (導出):

0 $G_0 = \{Q\}$.
1 Q に対して (H1) を使って: $G_1 = \{R, S\}$.
2 R に対して (H2) を使って: $G_2 = \{S, S\}$.
3 S に対して (H3) を使って: $G_3 = \{S\}$.
4 S に対して (H3) を使って: $G_4 = \{\}$.
5 ゴールが空集合になったので、実行は成功して終わる。

S が 2 回出てくるので、H4 が 2 回使われる点は、少し効率が悪いが、ともかく導出は成功する。この証明図は以下のようになる。

$$\dfrac{\dfrac{\dfrac{\overset{(H3)}{S} \qquad \overset{(H2)}{S \supset R}}{R}\supset\text{-E} \qquad \overset{(H3)}{S}}{R \wedge S}\wedge\text{-I} \qquad \overset{(H1)}{(R \wedge S) \supset Q}}{Q}\supset\text{-E}$$

例 3

```
H1   P :- Q, R.
H2   P :- S.
H3   Q :- S.
H4   S.
```

ゴール?- P. の実行 (導出):

0　$G_0 = \{P\}$.
1　P に対して、2 通りの可能性があるので、まずは (H1) を使って: $G_1 = \{Q, R\}$.
2　Q に対して (H3) を使って: $G_2 = \{S, R\}$.
3　S に対して (H4) を使って: $G_3 = \{R\}$.
4　R に対して、どの Horn 節のヘッドとも一致しないので失敗する。
5　先ほど試さなかった選択肢まで戻る。$G_5 = G_0 = \{P\}$.
6　P に対して (H2) を使って: $G_6 = \{S\}$.
7　S に対して (H4) を使って: $G_7 = \{\}$.
8　ゴールが空集合になったので、実行は成功して終わる。

この場合、H1 を使う場合を先にためし、それが失敗したら、H2 を使う場合をためした。Horn 節の「ヘッド」とは、`:-` の前にある原子論理式のことである。ただし、`P.` の形の Horn 節のヘッドは、`P` である。

3　演習問題

- 例 2 で、ゴール $G_0 = \{Q, S\}$ に対する導出を書きなさい。
- 例 2 のプログラムの先頭に、`P :- P.` という Horn 節を追加したとする。これは論理的には $P \supset P$ という論理式に対応するので、これを加えても、まったく問題なさそうである。しかし、これを追加すると、導出は停止しない。つまり、$G_0 = \{P\}$ というゴールに対して、Prolog の処理は停止しない。なぜか。
- 同様に、例 2 のプログラムの末尾に、`P :- P.` という Horn 節を追加したとする。なにが起きるか考え、なぜそうなるか、考えなさい。

4　まとめ

- 論理プログラミング: 「推論」過程を「計算」と見なす。
- Prolog: 一階述語論理のサブセットに対する効率良い推論手続き (導出原理) に基づくプログラミング言語
- 宣言型プログラミング vs 手続き型プログラミング (仕様 vs 実装)

A　述語論理の範囲の例

今度は、以下の Prolog プログラムを考える。

```
H1   r(X,Y) :- a(X,Y).
H2   r(X,Z) :- a(X,Y), r(Y,Z).
H3   a(tokyo,ibaraki).
H4   a(ibaraki,fukushima).
```

これらを、論理式として表現すると、以下の通りである。(ここでは Prolog 風に、変数を大文字で、述語記号等を小文字で書いた。)

$$\begin{array}{ll} H1 & \forall X \forall Y\ (a(X,Y) \supset r(X,Y)) \\ H2 & \forall X \forall Y \forall Z\ ((a(X,Y) \wedge r(Y,Z)) \supset r(X,Z)) \\ H3 & a(\mathrm{tokyo},\ \mathrm{ibaraki}) \\ H4 & a(\mathrm{ibaraki},\ \mathrm{fukushima}) \end{array}$$

この例では、項に対する変数 x, y, z が含まれるので、$\forall$ が現れている。

r(tokyo, ibaraki) の導出:

H5, H6, H7, H8 から $r(\mathrm{tokyo},\ \mathrm{ibaraki})$ を導く証明図は以下のようになる。

$$\dfrac{\stackrel{(H3)}{a(\mathrm{tokyo},\ \mathrm{ibaraki})} \qquad \dfrac{\stackrel{(H1)}{\forall X \forall Y\ (a(X,Y) \supset r(X,Y))}}{a(\mathrm{tokyo},\ \mathrm{ibaraki}) \supset r(\mathrm{tokyo},\ \mathrm{ibaraki})}\ \forall\text{-E}}{r(\mathrm{tokyo},\ \mathrm{ibaraki})}\ \wedge\text{-I}$$

これに対応する Prolog での導出を以下に図示する。