

関数・論理型プログラミング実験 第12回

江口 慎悟
酒寄 健
塚田 武志
松下祐介

講義のサポートページ

<http://www.kb.is.s.u-tokyo.ac.jp/~tsukada/cgi-bin/m/>

- 講義資料等が用意される
- レポートの提出先
- 利用にはアカウントが必要
- 名前/学籍番号/希望アカウント名をメールを
tsukada@kb.is.s.u-tokyo.ac.jp
までメールしてください。
 - 件名は「FL/LP実験アカウント申請」
 - アカウント名/パスワードを返信
 - PCからのメールを受け取れるように

論理型プログラミング（全3回）

第10回 Prolog の使い方

- Prolog を使ってみよう

第11回 手続き的側面

- 評価メカニズム

第12回 論理的側面

- 完全性・健全性
- 否定と閉世界仮説

今日の内容

○ Prolog の論理的側面

- Prolog プログラムの論理的解釈
- 否定と閉世界仮説
- SLD導出の完全性・健全性
- 実際の処理系との差異

Prolog プログラムの 論理的解釈

プログラムと問い合わせの意味

否定を扱うには

Prologの論理的解釈

- 問い合わせは（一階述語論理の）命題
 - `?- male(koji).` という問い合わせは `male(koji)` が証明できるかどうかを聞いている
- プログラムは命題の証明に使える仮定

```
male(koji).  
parent(kobo, koji).  
father(X, Y) :- parent(X, Y), male(Y).
```

- このプログラムは三つの仮定を記述している

問い合わせの意味

- 複数の問い合わせは 論理積 を取る

- 例 : ?- parent(kobo, koji), male(koji).
 \Rightarrow parent(kobo, koji) \wedge male(koji)

- 変数は存在量化する

- 例 : ?- parent(kobo, X).
 $\Rightarrow \exists X. \text{parent(kobo, X)}$

- 例 : ?- parent(kobo, X), male(X).
 $\Rightarrow \exists X. \text{parent(kobo, X)} \wedge \text{male(X)}$

プログラムの意味

- 変数は全称量化する

- 例: `positive(s(X)).`
 $\Rightarrow \forall X. \text{positive}(s(X))$

- ルールは「含意 (ならば) 」に対応

- 例: `father(X,Y) :- parent(X,Y), male(Y).`
 $\Rightarrow \forall X \forall Y. \text{parent}(X,Y) \wedge \text{male}(Y) \rightarrow \text{father}(X,Y)$

- プログラムはすべてのルール・事実の論理積

例

○ プログラム

```
male(koji).  
parent(kobo, koji).  
father(X, Y) :- parent(X, Y), male(Y).
```

■ 次の三つの仮定から成る

1. `male(koji)`
2. `parent(kobo, koji)`
3. $\forall X \forall Y. \text{parent}(X, Y) \wedge \text{male}(Y) \rightarrow \text{father}(X, Y)$

○ 問い合わせ

```
?- father(kobo, X).
```

- ### ■ $\exists X. \text{father}(\text{kobo}, X)$

証明してみよう

1. $\text{male}(\text{kobi})$
2. $\text{parent}(\text{kobo}, \text{kobi})$
3. $\forall X \forall Y. \text{parent}(X, Y) \wedge \text{male}(Y) \rightarrow \text{father}(X, Y)$

i) 仮定 2 から

$$\text{parent}(\text{kobo}, \text{kobi})$$

ii) さらに仮定 1 から

$$\text{parent}(\text{kobo}, \text{kobi}) \wedge \text{male}(\text{kobi})$$

iii) 仮定 3 の $X = \text{kobo}, Y = \text{kobi}$ の場合から

$$\text{father}(\text{kobo}, \text{kobi})$$

従って $\exists X. \text{father}(\text{kobo}, X)$

証明してみよう

1. $\text{male}(\text{kobi})$
2. $\text{parent}(\text{kobo}, \text{kobi})$
3. $\forall X \forall Y. \text{parent}(X, Y) \wedge \text{male}(Y) \rightarrow \text{father}(X, Y)$

i) 仮定 2 から

$\text{parent}(\text{kobo}, \text{kobi})$

ii) さらに仮定 1 から

$\text{parent}(\text{kobo}, \text{kobi}) \wedge \text{male}(\text{kobi})$

iii) 仮定 3 の $X = \text{kobo}, Y = \text{kobi}$ の場合から

$\text{father}(\text{kobo}, \text{kobi})$

従って $\exists X. \text{father}(\text{kobo}, X)$

証明

証明してみよう

1. $\text{male}(\text{kobi})$
2. $\text{parent}(\text{kobo}, \text{kobi})$
3. $\forall X \forall Y. \text{parent}(X, Y) \wedge \text{male}(Y) \rightarrow \text{father}(X, Y)$

i) 仮定 2 から

$\text{parent}(\text{kobo}, \text{kobi})$

ii) さらに仮定 1 から

$\text{parent}(\text{kobo}, \text{kobi}) \wedge \text{male}(\text{kobi})$

iii) 仮定 3 の $X = \text{kobo}, Y = \text{kobi}$ の場合から

$\text{father}(\text{kobo}, \text{kobi})$

従って $\exists X. \text{father}(\text{kobo}, X)$



SLD 導出

Prolog プログラムの 論理的解釈

プログラムと問合せの意味

否定を扱うには

復習：失敗による否定

- Prolog における否定は「失敗による否定」

- A の導出を試みて失敗したら、 $\neg A$ を導出する

否定の記号は Prolog では \neg

- 例

```
male(koji).  
parent(kobo, koji).  
father(X, Y) :- parent(X, Y), male(Y).
```

- female(koji) は導出できない。
よって $\neg \text{female(koji)}$ が導出できる

さきほどの解釈の問題点

- 否定的な命題を証明することはできない

```
male(koji).  
parent(kobo, koji).  
father(X, Y) :- parent(X, Y), male(Y).
```

1. male(koji)
2. parent(kobo, koji)
3. $\forall X \forall Y. \text{parent}(X, Y) \wedge \text{male}(Y) \rightarrow \text{father}(X, Y)$

閉世界仮説

- 「書かれていない命題は偽」とする仮定
 - 例

```
male(koji).  
parent(kobo, koji).  
father(X, Y) :- parent(X, Y), male(Y).
```

- $\neg \text{parent}(\text{koji}, \text{kobo})$
 - $\forall X. \neg \text{female}(X)$
- Prolog の否定の振舞いは、閉世界仮説でおおむね説明できる

もう少し複雑な例

```
male(kobo).  
male(koji).  
female(sanae).  
parent(kobo, koji).  
parent(kobo, sanae).  
father(X, Y) :- parent(X, Y), male(Y).
```

もう少し複雑な例

```
male(kobo).  
male(koji).  
female(sanae).  
parent(kobo, koji).  
parent(kobo, sanae).  
father(X, Y) :- parent(X, Y), male(Y).
```

$\forall X. (male(X) \leftrightarrow (X = kobo) \vee (X = koji))$

もう少し複雑な例

```
male(kobo).  
male(koji).  
female(sanae).  
parent(kobo, koji).  
parent(kobo, sanae).  
father(X, Y) :- parent(X, Y), male(Y).
```

$\forall X. (\text{male}(X) \leftrightarrow (X = \text{kobo}) \vee (X = \text{koji}))$

$\forall X. (\text{female}(X) \leftrightarrow (X = \text{sanae}))$

もう少し複雑な例

```
male(kobo).  
male(koji).  
female(sanae).  
parent(kobo, koji).  
parent(kobo, sanae).  
father(X, Y) :- parent(X, Y), male(Y).
```

$$\forall X. (\text{male}(X) \leftrightarrow (X = \text{kobo}) \vee (X = \text{koji}))$$
$$\forall X. (\text{female}(X) \leftrightarrow (X = \text{sanae}))$$
$$\forall X \forall Y. \left(\text{parent}(X, Y) \leftrightarrow (X = \text{kobo}) \wedge (Y = \text{koji} \vee Y = \text{sanae}) \right)$$

もう少し複雑な例

```
male(kobo).  
male(koji).  
female(sanae).  
parent(kobo, koji).  
parent(kobo, sanae).  
father(X, Y) :- parent(X, Y), male(Y).
```

$$\forall X. (\text{male}(X) \leftrightarrow (X = \text{kobo}) \vee (X = \text{koji}))$$
$$\forall X. (\text{female}(X) \leftrightarrow (X = \text{sanae}))$$
$$\forall X \forall Y. \left(\begin{array}{c} \text{parent}(X, Y) \leftrightarrow \\ (X = \text{kobo}) \wedge (Y = \text{koji} \vee Y = \text{sanae}) \end{array} \right)$$
$$\forall X \forall Y. (\text{father}(X, Y) \leftrightarrow \text{parent}(X, Y) \wedge \text{male}(Y))$$

手続きと論理の関係

SLD導出の完全性・健全性

Prologと論理の差

手続きと論理と

- 前回： Prolog の処理系の動作原理
 - Prolog の手続き的側面
- 今回： Prolog プログラムの論理的解釈
 - Prolog の論理的側面
- この二つの関係はどうなっているのか？

手続きと論理と

- 前回： Prolog の処理系の動作原理
 - Prolog の手続き的側面
- 今回： Prolog プログラムの論理的解釈
 - Prolog の論理的側面
- この二つの関係はどうなっているのか？
 - 原理的には二つは一致
 - しかし、実装上の都合で、異なる側面も

手続きと論理の関係

SLD導出の完全性・健全性

Prologと論理の差

SLD導出の完全性・健全性

- プログラム P と問合せ Q について以下は同値
 - P を使って SLD 導出によって Q が導出できる
(Prolog プログラムの手続き的解釈)
 - P の論理的解釈から Q の論理的解釈が証明できる
(Prolog プログラムの論理的解釈)
 - ゆえに手続き的解釈と論理的解釈は一致する
- ※ ただし、ここでいう SLD 導出では、
「ルールは上が優先」という規則は採用しない

手続きと論理の関係

SLD導出の完全性・健全性

Prologと論理の差

Prologと論理の差

- ルールとゴールの優先順位
- 単一化の出現チェック
- 変数のある命題の否定
- カットの有無
 - 論理に（Prolog的な）カットはない

ルールとゴールの優先順位

○ Prolog

- 上のルールが優先
- 左から右の順序でゴールを解決

○ 論理

- ルールに優先順位を考えない
 - いつでも、どのルールでも適用可能
- ゴールの解決の順序も任意

例

- 下のプログラムは、
 - Prolog 処理系に与えたときの挙動が異なる
 - 上は無限ループに陥る
 - 論理的解釈は同じ

```
ancestor(X,Y) :- ancestor(Z,Y), parent(X,Z).  
ancestor(X,Y) :- parent(X,Y).
```

```
ancestor(X,Y) :- parent(X,Y).  
ancestor(X,Y) :- parent(X,Z), ancestor(Z,Y).
```

単一化の出現チェック

- 多くの Prolog 処理系では、単一化において「出現チェック」を省いている

- パフォーマンスのため

- ※ 出現チェック：「 $x = t$ 」という制約は、 x に t を入れる代入が最汎単一化子になる。
ただし t 中に x が出現してはならない

- 出現チェックをさぼると、
証明できないのに導出される命題が生じる

例

```
add(z, Y, Y).  
add(s(X), Y, s(Z)) :- add(X, Y, Z).  
p :- add(s(z), X, X).
```

```
?- p.  
true.
```


例

```
add(z, Y, Y).
```

```
add(s(X), Y, s(Z)) :- add(X, Y, Z).
```

```
p :- add(s(z), X, X).
```

```
?- p.
```

```
true.
```

```
?- add(s(z), X, X).
```

```
X = s(X).
```

変数のある命題の否定

- 「失敗による否定」は
変数のない命題にだけ使える
 - 論理的に正しい「失敗による否定」の規則：
変数を持つ命題の否定は計算せずに、
別のゴールの導出を先に行う。
変数が具体化されるのを待って、否定を計算する
- Prolog の処理系では、変数の有無を気にせず、
「失敗による否定」規則を使おうとする

例

```
male(kobo).  
male(koji).  
female(sanae).  
parent(kobo, koji).  
parent(kobo, sanae).  
mother(X,Y) :- ¥+male(Y), parent(X, Y).
```

```
?- ¥+male(Y).  
false.  
?- mother(kobo, Y).  
false.  
?- mother(kobo, sanae).  
true.
```

例

```
male(kobo).  
male(koji).  
female(sanae).  
parent(kobo, koji).  
parent(kobo, sanae).
```

parent(X,Y), $\text{!male}(Y)$
なら期待通りの動作

```
mother(X,Y) :-  $\text{!male}(Y)$ , parent(X, Y).
```

```
?-  $\text{!male}(Y)$ .  
false.  
?- mother(kobo, Y).  
false.  
?- mother(kobo, sanae).  
true.
```

例題

理解の確認をするための課題です

課題提出システム上での提出の必要はありません

例題を解きTAに見せることで出席とします

分からないことがあったら、積極的に質問しましょう

例題

○ 次を意味するプログラムを書け

1. $\forall X. \text{sub}(X, z, X).$
2. $\forall X \forall Y \forall Z. \text{sub}(X, Y, Z) \rightarrow \text{sub}(s(X), s(Y), Z).$

○ 以下のそれぞれの論理式に相当する
問い合わせをせよ

- $\exists X. \text{sub}(s(s(s(z))), s(z), X)$
- $\exists X. \text{sub}(z, s(z), X)$

レポート課題12

締切：2019/7/23 13:00(JST)

問 1

○ 次のプログラムを考える

```
eq(a, b).  
eq(c, b).  
eq(X, Z) :- eq(X, Y), eq(Y, Z).  
eq(X, Y) :- eq(Y, X).
```

- 論理的解釈では $\text{eq}(a, c)$ が true であることを示せ
- Prolog 処理系で `?- eq(a, c).` を
問い合わせるとどうなるか。それはなぜか
- 処理系をどう工夫すれば、この差が埋まるか

参考：プログラムの論理的解釈

```
eq(a, b).  
eq(c, b).  
eq(X, Z) :- eq(X, Y), eq(Y, Z).  
eq(X, Y) :- eq(Y, X).
```

1. $eq(a, b)$
2. $eq(c, b)$
3. $\forall X \forall Y \forall Z. eq(X, Y) \wedge eq(Y, Z) \rightarrow eq(X, Z)$
4. $\forall X \forall Y. eq(Y, X) \rightarrow eq(X, Y)$

eq は部分同値関係 (partial equivalence relation)

問 2

○ 次のプログラムを考える

```
test :- q(X, X).  
      q(X, f(X)).
```

- 論理的解釈では ?- test. の問合せの結果は
どうなると考えられるか
- 実際に ?- test. を Prolog処理系に問い合わせると
どうなるか。どうしてそうなるのか。
 - ヒント : ?- q(X, X). を問い合わせるとどうなるか

問 3

- 以下のプログラムと問合せを考える。

```
p(a).  
q(b).
```

```
?- ∀+p(X), q(X).
```

- 論理的な解釈から期待される結果は何か
- Prolog 処理系に実際に問い合わせるとどうなるか。
なぜそのような結果になるか。
- 論理的解釈と Prolog 処理系の応答が一致するように
問合せを書き換えよ

参考：プログラムの論理的解釈

$p(a).$
 $q(b).$

1. $\forall X. (p(X) \leftrightarrow X = a)$
2. $\forall X. (q(X) \leftrightarrow X = b)$

発展 1

○ 次のプログラムと問合せを考える。

```
r(a) :- p(a).  
r(a) :- ✕+p(a).  
p(X) :- p(f(X)).
```

```
?- r(a).
```

- $r(a)$ がプログラムの論理的帰結であることを示せ
- Prolog処理系はこの問合せにどう答えるか。
それはなぜか。

発展 2

- 論理的意味に対して完全かつ健全な Prolog ライクな論理型言語を実装せよ
 - すべての解代入が、いずれ出力される
 - 同じ解代入を何度出力してもよい
 - すべての解代入を出力しても、停止する必要はない
 - 構文解析器を準備する必要はない
 - カットや否定はなくてよい