

関数・論理型プログラミング実験 第10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

Prolog 処理系の導入

- `apt-get install swi-prolog`
- `aptitude install swi-prolog`

Prolog?

- 論理プログラミング言語
 - 一階述語論理に基づく宣言的プログラミング言語
- 非決定性計算 + build-in search
 - generate-and-test プログラミング

今日の内容

- 使ってみよう
- 構文の概要
- もう少し複雑な例

使ってみよう

family.pl

```
/* family.pl */  
male(kobo).  
male(koji).  
male(iwao).  
female(sanae).  
female(mine).
```

kobo は male であるという
fact の宣言

ピリオドが区切り

起動、読み込み

```
$ swipl
```

```
...
```

```
...
```

```
?- ['family.pl'].
```

```
% family.pl compiled 0.00 sec, 6 clauses  
true.
```

```
?-
```

問い合わせ

```
?- male(kobo).  
true.
```

```
?- female(sanae).  
true.
```

```
?- female(kobo).  
false.
```

問い合わせ（続）

```
?- male(X).
```

male であるような X は？

```
X = kobo ;
```

```
X = koji ;
```

```
X = iwao.
```

「;」 他には？
(tabキーでも同じ)

```
?- male(X).
```

```
X = kobo .
```

```
?-
```

「.」 で終了
(Enterキーも同じ)

より複雑な例：親子関係

```
/* family.pl */  
male(kobo).  
male(koji).  
male(iwao).  
female(sanae).  
female(mine).
```

「koboとkojiは
parentという関係にある」
というfactの宣言

```
parent(kobo, koji).  
parent(kobo, sanae).  
parent(sanae, iwao).  
parent(sanae, mine).
```

使ってみる

```
?- ['family.pl'].
```

```
% family.pl compiled 0.00 sec, 10 clauses  
true
```

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

```
X = koji ;
```

```
X = sanae.
```

```
?- parent(X, iwao).
```

```
X = sanae.
```

父親、母親

```
/* family.pl */
```

(…略…)

```
father(X, Y) :- parent(X, Y), male(Y).  
mother(X, Y) :- parent(X, Y), female(Y).
```

「XとYがfatherなのは、
XとYがparent関係にあり、
Yがmaleのとき」
というruleの宣言



(右から左への含意)

問い合わせ例

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

```
X = sanae.
```

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

```
X = koji ;
```

```
false.
```

注：prolog はこの時点では
他のfather候補も持っているので
続きを調べるか聞いてくる

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

```
X = koji ;
```

```
X = iwao ;
```

```
false.
```

「_」はワイルドカード

祖父母、祖先

```
/* family.pl */
```

```
(...略...)
```

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

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

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


問い合わせ例

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

```
X = iwao ;
```

```
X = mine.
```

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

```
X = koji ;
```

```
X = sanae ;
```

```
X = iwao ;
```

```
X = mine ;
```

```
false.
```

```
?- grandparent(kobo, X), male(X).
```

```
X = iwao ;
```

```
false.
```

注意：述語やruleの順番

- 上から下の順でルールを当てはめようとする
- 左から右の順でチェックする

はじめに `parent(X,Z)` か調べる

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

次に、`parent(X,Y)`であるようなYを探し、
それが`ancestor(Y,Z)`を満たすか調べる

注意：述語やruleの順番

- 上から下の順でルールを当てはめようとする
- 左から右の順でチェックする
 - したがって、次は動かない

ancestor(X,Z)を調べるために
ancestor(Y,Z)を満たすYを探し、
そのためにancestor(Y1,Z)を満たすY1を探し、
...

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

否定

○ 難しいので、後回し（第12回）

- 通常の論理における否定とは、振る舞いが異なる
 - 閉世界仮説
 - Negation as Failure

これまでのまとめ

- .pl に fact や rule を書く
- インタプリタで問い合わせができる
 - father(kobo, X). のような問い合わせを行うと、prolog が X を計算してくれる

構文の概要

コメント

○ C風

- `/* comment */`
- ネストできる

項

○ 定数

- kobo, koji, sanae, ...
- 整数
- 文字列 'hello world'

○ 変数

- X, Y, Ab, _C, ...

○ 複合項

- $f(\text{複合項}, \dots, \text{複合項})$
 - f を functor と呼ぶ
 - f の構文は整数でない定数と同じ
- $s(z), s(s(z)), \dots$ など

fact

○ 述語(項, ..., 項).

```
male(kobo).
```

```
male(koji).
```

```
male(iwao).
```

```
female(sanae).
```

```
female(mine).
```

```
parent(kobo, koji).
```

```
parent(kobo, sanae).
```

```
parent(sanae, iwao).
```

```
parent(sanae, mine).
```

rule

○ 述語(項, ..., 項) :- 述語(項, ..., 項), ..., 述語(項, ..., 項).

```
/* family.pl */
```

```
(...略...)
```

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

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

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

リスト記法

- $[t_1 | t_2]$
 - コンス
 - $[]$
 - 空リスト
 - $[1, 2, 3]$
 - $[1 \mid [2 \mid [3 \mid []]]]$ のこと
 - $[t_1, t_2 \mid t_3]$
 - $[t_1 \mid [t_2 \mid t_3]]$ のこと
- (Prologは型なしなので $[[1], 2]$ も可)

もう少し複雑な例

加算

```
/* add.pl */  
add(z, Y, Y).  
add(s(X), Y, s(Z)) :- add(X, Y, Z).
```

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

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

(この定義では) 減算もできる

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

注意

- どれを変数にして問い合わせできるかは定義次第

```
/* add.pl */
```

```
...
```

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

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

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

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

```
ERROR: Out of local stack
```

```
...
```

リストの連接

```
/* app.pl */  
append([], Y, Y).  
append([A|X], Y, [A|Z]) :- append(X, Y, Z).
```

```
?- append([1,2,3], [4,5], Z).  
Z = [1,2,3,4,5].
```

```
?- append([_,_], Y, [1,2,3,4,5]).  
Y = [3,4,5].
```

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

- コボには、実は妹のミホがいる
この事実に合わせて family.pl を拡張せよ
- 兄弟姉妹を表す関係 sibling を定義せよ
 - sibling(X, X) が成り立っても良いとする
- 定義した sibling の動作を確認せよ

レポート課題10

締切：2019/7/9 13:00(JST)

問 1

- X と Y が血縁関係を表す二項述語
bloodrelative を定義せよ
 - 血縁：共通の祖先を持つこと
- 定義した述語の動作を適当な例で確認せよ

問 2

- $s(s(\dots(s(z))\dots))$ の形式で与えられる整数の積を計算する三項述語 `mult` を定義せよ

$$\text{mult}(X, Y, Z) \Leftrightarrow X * Y = Z$$

- 適当な例について動作を確認せよ
 - どれを変数にして問い合わせができるか

問 3

- X と Y が逆順のリストであることを表す二項述語 reverse を定義せよ
 - ただし次の通りになるように

```
?- reverse([1,2,3], X).  
X = [3,2,1]
```

- X の各リストを順に接続したものが Y であることを表す二項述語 concat を定義せよ

```
?- concat([[1], [2,3]], X).  
X = [1,2,3]
```

V は重複がないと仮定してよい、

問 4

- V を頂点のリスト、
 E を枝のリストとしたとき
有向グラフ (V, E) がハミルトン路を持つ
ことを表す述語 $\text{hamilton}(V, E)$ を定義せよ
- 「路」であって「閉路」でないことに注意
(閉路の方が、やや難しい?)

発展 1

- 今日紹介した構文の範囲で
Prolog はチューリング完全である。
このことを示せ。
- きちんとした意味論は与えていないので、
証明のスケッチや戦略でよい