プログラム言語論

亀山幸義

筑波大学 情報科学類

論理型プログラム言語

論理型プログラム言語とは

- 「推論＝計算」という考えで設計された言語.
- プログラムは，事実，あるいは，推論規則 (事実から事実を導く) として記述.
- どのように推論するかの手順は，記述しない.
- 宣言型プログラム言語 (Declarative Programming Language) の一種.

Prolog プログラミング-1

```
is_mother(alice, charlie).
is_mother(alice, eliza).
is_mother(eliza, george).
is_mother(hillary, fritz).
is_husband(bob, alice).
is_husband(fritz, eliza).
is_father(X, Z) :- is_husband(X, Y),  is_mother(Y, Z).
is_parent(X, Y) :- is_mother(X, Y).
is_parent(X, Y) :- is_father(X, Y).
is_grandparent(X, Z) :- is_parent(X, Y), is_parent(Y, Z).
is_spouse(X, Y) :- is_husband(X, Y).
is_spouse(X, Y) :- is_husband(Y, X).
is_relative(X, X).
is_relative(X, Y) :- is_spouse(X, Y).
is_relative(X, Y) :- is_ancester(X, Y).
is_relative(X, Y) :- is_ancester(Y, X).
```


Prolog プログラミング-2

```
?- is_mother(X, charlie).
X = alice

?- is_mother(alice, X).
X = charlie n
X = eliza.

?- is_husband(alice, X).
false.

?- is_father(bob, X).
X = charlie n
X = eliza.
```

Prolog プログラミング-3

```
?- is_parent(X,eliza).
X = alice n
X = bob n
false.

?- is_grandparent(X,Y).
X = alice,
Y = george n
X = bob,
Y = george n
X = bob,
Y = george n
false.
```


Prolog プログラミング-4

```
?- is_relative(alice,X).
X = alice n
X = bob n
X = charlie n
X = eliza n
X = george n
false.

?- is_relative(eliza, X).
X = eliza n
X = fritz n
X = george n
X = alice n
X = bob n
false.
```


Prolog プログラミング-5

解が無限にたくさんある例。

```
% the following code is buggy.
is_superrelative(X, Y) :- is_relative(X, Y).
is_superrelative(X, Z) :- is_relative(X, Y), is_superrelative(

?- is_superrelative(alice, X).
X = alice n
X = bob n
X = charlie n
X = eliza n
X = george n
X = alice n
X = bob n
X = charlie n
X = eliza n
...
```


Prolog プログラミング-6

```
add(X, 0, X).
add(X, s(Y), s(Z)) :- add(X, Y, Z).
times(X, 0, 0).
times(X, s(Y), W) :- times(X, Y, Z), add(Z, X, W).
power(X, 0, s(0)).
power(X, s(Y), W) :- power(X, Y, Z), times(Z, X, W).
fact(N, 1) :- N < 1, ! .
fact(N, X) :- N1 is N - 1, fact(N1, X1), X is X1 * N .
append([], X, X).
append([X | L], M, [X | N]) :- append(L, M, N).
?- fact(10, X).
X = 3628800.
?-  append([a, b, c], [d, e, f], X ).
X = [a, b, c, d, e, f].
```

Prolog プログラミング-7

`append(X, Y, [a, b, c, d, e, f])` とやると何が返ってくるだろうか？

Prolog プログラミング-8

```
?- append(X, Y, [a, b, c, d, e, f]).
X = [],
Y = [a, b, c, d, e, f] n
X = [a],
Y = [b, c, d, e, f] n
X = [a, b],
Y = [c, d, e, f] n
X = [a, b, c],
Y = [d, e, f] n
X = [a, b, c, d],
Y = [e, f] n
X = [a, b, c, d, e],
Y = [f] n
X = [a, b, c, d, e, f],
Y = [] n
false.
```


Prolog プログラミング

宣言型プログラム言語 (Declarative Programming Language) の一種.

- 推論＝計算.
- プログラム＝事実，あるいは，推論規則.
- ゴール＝その事実が成立するかどうかを質問.
- 推論をする手順は，記述しない.

特徴.

- プログラムの目的=ゴールが成立するかどうか，を求める.
- 求解＝ゴール中の変数のある値に対して，ゴールが成立するか？
- 複数の解があることもある.
- あらゆる可能性を網羅的に探索する．双方向計算が可能.
- 一階述語論理のサブセット (Horn 節論理) に対応.

論理型プログラム言語

- Kowalski 1974 が提唱.
- Colmerauer 1973 が Prolog の最初の処理系.
- 日本の ICOT(第 5 世代コンピュータ・プロジェクト) が大きな貢献.
- 現在でも，知識表現，帰納論理プログラミングなどの分野で活躍.

Prolog = Programming in Logic

Prolog 処理系

ここでは Linux 上の SWI-Prolog 処理系を使った。
(http://www.swi-prolog.org/)

```
% swipl
Welcome to SWI-Prolog (Multi-threaded, 32 bits, Version 5.6.58
Copyright (c) 1990-2008 University of Amsterdam.
...

For help, use ?- help(Topic). or ?- apropos(Word).

?- [test-prolog].   (←ここで test-prolog.pl ファイルの読み込み)
[test-prolog].
...
% test-prolog compiled 0.00 sec, 5,280 bytes
true.

?-  append(X, Y, [a, b, c, d, e, f]).
X = []
```


まとめ

- 「推論」を計算過程と見なしたプログラミング言語.
- その他にも様々な (驚くような) 仕組みを計算の原理に使ったものがあり得る。
  - 量子コンピューティング, DNA コンピューティング, ...

Quiz

「事実やルール (推論規則) を記述するだけで，プログラムとして動く」という言語は，メリットもデメリットもある.

- メリット: どのルールをどう使ったら，ゴールとなる事実を導けるか (効率的に導くにはどうしたらよいか) を，プログラマは考えなくてよい．解決したい問題を正確に記述するだけでよい．(プログラミングが楽になる．)
- デメリット: 常に Prolog 処理系が決めた順序で解を探索するので，効率が必ずしも良くない．(効率を良くしようと思うと，処理系の動きを把握していないといけない．)

このような言語が，現実に有用な場面としてどんなものがあるか、想像して書きなさい.
参考: 「プログラムの仕様」がそのまま動くものを、executable specification(実行可能な仕様) という。