ソースプログラムと実行例

まず，以下の説明に従って Prolog の動きを試してみる．

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  sample program (sample.pl)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

エディタを開いて下の 3 行 (プログラム) を打ち込み
名前をつけて (たとえば sample.pl) 保存する．
（拡張子は pl，ファイル名は ASCII code.）
一種のデータベース，ルールベースを定義したと思えばよい．
これがソースプログラムである．

% 以下には各行の意味が書かれている．
これらはコメントなので打ち込む必要はない．

```
----- ソースプログラム -----------------
fr(apple).   % りんごは果物である
fr(banana).  % バナナは果物である
sweet(X) :- fr(X). % (任意の X について) X が果物であるならば X は甘い
---------------------------------------
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  sample session
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

ソースプログラムには述語の定義が書かれており，
直観的には，プログラムを実行することで，ある命題の真偽値やその命題を真にする値を求める．

ファイルを SWI-Prolog のアイコンに drag&drop することで
sample.pl をロードする．
（実習では Prolog はコンパイラではなくインタプリタで行う.）

?- が入力待ちプロンプトであり，この後ろにキーボードから入力する．
入力の最後にはピリオドをつけて Enter キーを押す．

まず，fr という述語の定義を確認する．
（単なる確認なのでやらなくてもよい.）
すると，2 つの定義が表示される．
（true はこの述語が正しいことを示す.）

```
1 ?- listing(fr).

fr(apple).
fr(banana).
true.
```

次に,「りんごは果物か？」を質問する．
すると，true (正しい) という答えが返ってくる．
りんごはデータベースに登録されているからである．

```
2 ?- fr(apple).
true.
```

次に,「バナナは果物か？」を質問する．
すると，true (正しい) という答えが返ってくる．

```
3 ?- fr(banana).
true.
```

次に,「ぶどうは果物か？」を質問する.
すると，false（正しくない）という答えが返ってくる.
ぶどうはデータベースに登録されていないからである.

```
4 ?- fr(grape).
false.
```

次に,「果物はどういうものがあるか？」と質問する.
大文字から始まる引数は変数であり，X にはまず apple が返ってくる.

```
5 ?- fr(X).
X = apple .
```

ここで Enter を押して再び,「果物はどういうものがあるか？」と質問する.
今度は X=apple のあと Enter をおさずに ;（セミコロン）を押す.
すると別解が探索され，X=banana が返ってくる.

```
6 ?- fr(X).
X = apple ;
X = banana.
```

次に,「りんごは甘いか？」と質問する.
すると，第 3 節に書かれた規則が適用されて true が返ってくる.

```
7 ?- sweet(apple)
true.
```

セッションを終了する.
halt と打ち込むかまたは window を閉じる.

```
8 ?- halt.
```

1 Prolog プログラムの構造と意味

プログラムの構造

プログラムは有限個のホーン節 (Horn clause) から成る

- ホーン節の形

$$\underbrace{H}_{\text{頭部 (head)}} \text{ :- } \underbrace{B_1, \ldots, B_n.}_{\text{本体 (body)}}$$

  $H, B_1, \ldots, B_n$ をそれぞれ**ゴール (goal)** と呼ぶ．ゴールはアトムである．
  すなわち，述語記号 (引数$_0, \ldots,$ 引数$_m$) の形で記述される．

- ホーン節の論理的意味

$$B_1 \wedge \ldots \wedge B_n \rightarrow H$$

- ホーン節の種類
  - **単位節 (unit clause)** – 事実 (fact) を表す
    記述形式　$goal.$
    例　`parent(tom,liz).`
  - **確定節 (definite clause)** – ルール (rule) を表す
    記述形式　$goal$:- $goal_1, \ldots, goal_n.$
    例　`father(N,M) :- parent(N,M), male(N).`
  - **ゴール節 (goal clause)** – 質問 (query) を表す
    記述形式　:- $goal_1, \ldots, goal_n.$
    例　`:- parent(X,liz).`

「命題」とは「真」か「偽」かが決まる言明 (文) のことである。「述語 (ゴール)」は述語記号と 0 個以上の引数で表される命題である。

- 述語の例：「りんごは甘い」「X は Y の父親である」「X が果物であるならば X は甘い」「X の 2 乗は Y である」
- オブジェクトの例：「りんご」「私の父」「X の 2 乗」
- 述語でもオブジェクトでもない例：「X に 2 を入力する」「リストの最初の要素を取り出す」

プログラムの実行

上から下，左から右へと行われる
Prolog 処理系は大文字小文字を区別して扱う

- 変数は大文字または _ から始まる文字列
- 定数，関数記号，述語記号は小文字から始まる文字列

再帰プログラミング

ある対象を定義するのにその対象自身を使用することを再帰 (recursion) といい，そのような定義を再帰的定義という．Prolog の基本的なプログラミングスタイルであり，計算機科学における基本的かつ重要な考え方の 1 つである．

例 1 述語 ancestor(X,Y) 「X は Y の祖先である」の再帰的定義

ancestor(X,Y) ← parent(X,Y).

X が Y の親ならば X は Y の祖先である

ancestor(X,Y) ← parent(X,Z) ∧ ancestor(Z,Y).

X が Z の親でかつ Z が Y の祖先ならば X は Y の祖先である

例 2 述語 sum(X,Y) 「0 から X までの和は Y である」の再帰的定義

sum(0,0).

0 から 0 までの和は 0 である

sum(N,M) ← sum(N-1,M1) ∧ M=M1+N.

0 から N までの和 M は 0 から N-1 までの和 M1 に N を加えたものである

注意：これらの例は文法を一部変更しているのでプログラムそのものではない．

演習問題 (r1)

(1) 図 1.1 のデータベースにおいて tom, bob, jim, pam, liz, pat の年齢がそれぞれ 80, 58, 2, 76, 55, 27 であるという条件が加わったとする．このとき，まず，X の年齢が A であるという関係を表す述語 age(X,A) を使って各々の年齢を age(tom,80) のように記述することでデータベースに追加せよ．次にこの述語を使って X が Y よりも年上である (同い年は含まない) ことを表す述語 senior(X,Y) を定義せよ．大小関係を表すオペレータは $>, <, >=, =<$ 等ですべて中置である（教科書 p.88 参照）．

(2) right_of(Obj1,Obj2), above(Obj1,Obj2) が，Obj1 が Obj2 のすぐ右隣，真上にある関係をそれぞれ表すとする．図 1.2 のオブジェクトの配置を right_of(Obj1,Obj2), above(Obj1,Obj2) を使って記述せよ．また，left_of(Obj1,Obj2), below(Obj1,Obj2) が，Obj1 が Obj2 のすぐ左隣，真下にある関係をそれぞれ表すとするとき，これらを right_of, above を使って定義せよ．オブジェクトの名前はそれぞれ bicycle, bird, camera, apple, pencil, sandglass, butterfly, fish とする．

(3) (2) において，Obj1 が Obj2 のちょうど斜め右上にある関係 upper_right(Obj1,Obj2), 同行の右の方向にある関係 right(Obj1,Obj2) をそれぞれ定義せよ．right については**再帰的に**定義すること．

(4) 講義名，曜日，時限を引数として持つ述語を lecture とし，自分の今学期の時間割データベースを作成せよ．また，木曜日にある講義名を調べる述語 thursday(X) を定義せよ．さらに，木曜 5 限に知識情報処理および金曜 4 限に知識情報処理実習がはいっていれば真であることを表す 0 引数の述語 correct を定義せよ．

(5) edge(N,M) が有向グラフにおいてノード N から ノード M へのエッジがあるという関係を表すとする．edge を使って図 1.3 のようなグラフを表せ．
また，connected(N,M) をノード N からノード M へ直接/間接に行く経路が存在するという関係を表すとする．connected を**再帰的に**定義し， connected(X,Y) を実行して全解を確かめよ．ただし，2 通りの経路がある場合は 2 つ同じ解が得られる．

(6) 練習問題 1(4) の解答例の論理的意味を示せ．

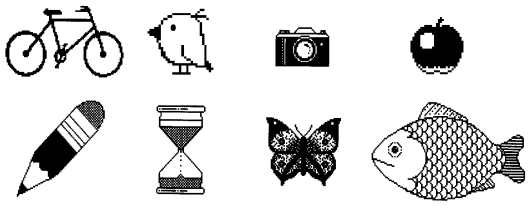

図 1.2

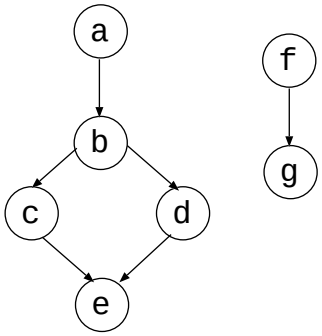

図 1.3

練習問題

1. parent(X,Y) を X が Y の親である，という関係を表し，male(X), female(X) がそれぞれ X は男性である，女性である，を表すとする．tom, bob, jim が男性，pam, liz, pat が女性で，彼らが 図 1.1 で表されるような親子関係を持つとき，parent, male，female 関係の成り立つものをすべて記述せよ．ただし，図で矢印は親から子をさしている．

   (1) これらを記述したデータベースを作成せよ．

   (2) tom が bob の親でかつ tom が liz の親であるならば正しい (ok) という関係を表す (0 引数の) 述語 ok を定義せよ．

   (3) X が Y の父親であるという関係を表す述語 father(X,Y) を parent, male を使って定義せよ．

   (4) X が Y の祖父であるという関係を表す述語 grandfather(X,Y) を定義せよ．必要ならば新たな変数を導入せよ．

   (5) X が Y の祖先であるという関係を表す述語 ancestor(X,Y) を定義せよ．必要ならば新たな変数を導入せよ．

   (6) (3) のプログラムの論理的意味を示せ．**命題の形になっていること**，すなわち，引数への入出力を書くのではなく,「C である」「A かつ B ならば C である」のように記述すること．

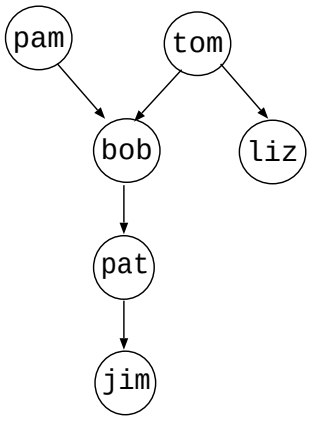

図 1.1