1 Prologue: Prologを動かすには

まず，なにはともあれPrologを動かす方法を学ぼう．ここで用いるPrologはSWI-Prologである．

1.1 シェルでの起動

SWI-Prologを起動する時には次のように，prologとタイプしてから改行キー (↵) を打つ．

```
% prolog ↵
```

すると，次のようなメッセージか返ってくる（はずである）[1] ：

```
Welcome to SWI-Prolog (Multi-threaded, Version 5.6.8)
Copyright (c) 1990-2006 University of Amsterdam.
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to redistribute it under certain conditions.
Please visit http://www.swi-prolog.org for details.

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

?-
```

この?- は，Prologの（トップレベル・）プロンプト (prompt) である．それは，「ユーザからの入力を受理する用意ができていること（つまり，入力待ちであること）」を表している．

注 1.1 SWI-Prolog関係のファイルは，ディレクトリ
/usr/local/lib/pl-5.6.8/
にある (計算機環境のupdateにより, 変更されるときがある．その場合は講義時に指示する).

次に簡単な実行例を示す.

[1] もし動かないときは，pathの設定がおかしい場合がある．ちゃんと動いている人の.cshrcファイルを参考にして設定して下さい.

```
?- write(hello).  ↵                % ユーザの入力
hello                              % システムの出力

yes                                % システムの出力
?- X is 2+2.  ↵                    % ユーザの入力

X = 4 ↵                            % (*)

yes                                % システムの出力
?-                                 % システムの出力
```

ここで，write や is は Prolog にあらかじめ用意されている組み込み (built-in) 述語である．

注 1.2 上の入力例に示すとおり，それぞれの入力の最後のピリオド . の後には改行を入力している．以下の例でも同様であるので，これ以降はこの改行記号↵は省略する．

また，上の (*) においては，X = 4 がシステムの出力で，改行記号↵はユーザの入力である．

1.2 Emacs での起動

Emacs において Prolog を起動することもできる．その際は，Emacs で

M-x run-prolog

とタイプする[2]．この場合は，実行結果が通常のファイルと同様に編集できるので，レポート作成時には便利である．

1.3 ファイルからのプログラムの読み込み

Prolog のプログラムを作る時は，簡単なプログラムの場合は直接入力してもよいが，そうでない場合は，通常エディタを用いてプログラムのファイルを作る．短いプログラムならば一つのファイルで，大規模なプログラムになると複数個のファイルを用いて作る．ここではエディタの使い方には，すでに熟知しているものとする．

Prolog のプログラムをファイルに書いておいてそれを読み込んで実行するには次のようにする：

```
['ファイル名'].
```

[2] M-x は，エスケープキーを押してから x をタイプすることを表す．もしこれで動かないときは，.emacs ファイルの設定がおかしい場合がある．ちゃんと動いている人のファイルを参考にして設定して下さい．

例 1.1 例えば，次のようなプログラムを書いて，ファイル mymember.pl[3] にセーブしたとする．

```
% member(X,List): X はリスト L の要素である
member(X, [X|_]).
member(X, [_|L]) :- member(X, L).
```

注 *1.3* 上の 2 行目の最後の．（ピリオド）の後に，改行（↵）を入れておく習慣をつけておいた方が良い．改行を入れ忘れると，(Prolog の処理系によっては）プログラムがうまく読み込まれない場合がある．

注 *1.4* 上で，_（underscore）と記されているのは，無名変数 (anonymous variables) である．その意味は，「その節にすで出現している変数以外ならば，どんな名前でも構わない，従ってわざわざ名前をつける必要がないという変数」を表す．従って上のプログラムは，例えば次のプログラムと同じ意味である．

```
member(X, [X|L]).
member(X, [H|L]) :- member(X, L).
```

注 *1.5* 上のプログラムで %以降の行はコメントと解釈される．

この mymember.pl というファイルを読み込むには，次のようにする．

```
?- [mymember].              %  ファイルの読み込み
% member compiled 0.00 sec, 1,152 bytes

Yes                         %  ファイルの読み込み完了
?- member(a,[a,b,c]).       %  ゴールの入力
                            %  ゴールの成功
Yes
?- member(d,[a,b,c]).       %  別のゴールの入力

No                          %  ゴールの失敗
?-                          %  次の入力待ち
```

1.4 実行の終了

Prolog の処理系から抜ける時は，^D（コントロールキーを押しながら D）をタイプする．

[3] このように，prolog プログラムのファイルは○○○.pl というように，ファイルの拡張子を pl にするのが普通である．Perl と同じ!!

```
?- ^D
```

あるいは，halt. の後改行キー↵をタイプする．

```
?- halt.  ↵
```

上のいずれの方法でも Prolog から抜けられない非常事態になった時は，^C（コントロールキーを押しながら C）をタイプし，その後 a ↵と打つとプロンプトに戻る（e ↵と打った場合は Prolog 自体を終了しシェルに戻る）．

```
?- ^C
Action (h for help) ?  a ↵
ERROR: Execution Aborted
% Execution Aborted
?-
```

1.5 Prolog の実行命令 (directive)

Prolog の実行は次のような**ゴール**（goal，あるいは**問合せ** (query) とも言う）を与えることによって行なう．

```
?- write('hello again').  ↵          % ゴールの入力
hello again                          % システムの出力
yes                                  % システムの出力
?-
```

先に定義した述語 member の実行を再び行なう：

```
?- member(b, [a,b,c]).      % ゴールの入力

yes
?-
```

これは，ゴールの実行が成功したこと（yes），そして次の入力待ち状態にあること（?- ）を示す．

次に，問合せが変数を含む場合を試してみると：

```
?- member(X, [a,b,c]).      % ゴールの入力

X = a ?
```

先にも説明した通り，この時点で，インタプリタはユーザからの次の二通りの入力を待っている状態にある．すなわち：

(i) 改行を入力すると
… このときは，これで実行終了．処理系は次のメッセージを出力する：

```
yes
?-
```

(ii) セミコロン (;) と改行を入力すると
… このときは，処理系は**バックトラック**（backtrack）して，別の解を探す．もし別解がない場合は no を返す．

```
?- member(X, [a,b,c]).

X = a ? ;

X = b ? ;

X = c ? ;

no
?-
```

1.6 ファイルの読み込み時の注意

前に次のプログラムをファイル mymember.pl にセーブした．

```
member(X, [X|_]).
member(X, [_|L]) :- member(X, L).
```

そして，このプログラムを “mymember.pl” という名前のファイルにセーブしたとしよう．そのファイルを次のようにして読み込んだ：

```
?- [mymember].
```

この場合，次のようにしてもファイルを読み込むことができる：

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

このように，Prolog 処理系にファイルを「読み込む」ことを，ファイルを「コンサルトする *consult*」という．

注 1.6 ［よくやる間違い］ ファイル名として，mymember.pl のように英数字以外の文字（この場合は'.'（ピリオド））を含んでいる場合は，そのファイル名を引用符（シングル・クォート）で囲まなければならない．そうしないと，次のようなエラーメッセージが表示され，プログラムが読み込まれない．

```
?- [mymember.pl].
ERROR: Syntax error: Operator expected
ERROR: [mymember
ERROR: ** here **
ERROR: .pl] .
?-
```

もし，プログラムがいくつかのファイルを必要とする場合は，例えば，次のようにする：

```
?- [hajime,tsugi,saigo].
```

これによって，三つのファイル hajime.pl，tsugi.pl，saigo.pl がすべて読み込まれる．

1.7 実行の中断

プログラムの実行の途中で，その実行を中断したくなった時（例えば無限ループに入ってしまった時など）は，^C（コントロールキーを押しながら C を打つ）をタイプすることによって，中断する (interrupt) ことができる．この時，次のようなメッセージが表示される：

```
Action (h for help) ?
```

この時点で，処理系は次のような 1 文字のコマンドを受け付ける．従って，ユーザは以下のコマンドの意味に応じて，1 文字（とその後に改行）をタイプする：

a	現在の計算を中止（アボート，abort）する．
b	現在の計算を中断（ブレーク，break）し，別のサブプロセスを呼び出す．
c	現在の計算を継続する．
g	現在計算しているゴールを表示する．
e	SWI-Prolog から抜ける．
h	利用可能なコマンドを表示する．
t	トレース（trace）を動かす．