● 組み込み述語

組み込み述語は、ライブラリモジュールと異なり、モジュール名を指定しないで呼び出します。

<alt 述語...>

省略可能な述語実行を示します。内部で呼び出された述語の実行結果に関わらずに必ずTrueで成功する述語です。
"[述語...]"とも書けます。
EBNF記法の[]として構文解析で使います。

<assert 節>
<asserta 節>
<assertz 節>

プログラムの１行の単位である節を追加します。
assertaは先頭に追加し、assertzは最後に追加します。
assertはassertaと同じで先頭に追加します。

<cd パス>

カレントディレクトリを引数で指定したパスに移す。

<edit ファイル名>
環境変数DEDITORPATHまたはEDITORに設定されたエディタを使ってファイル名を編集します。

<dir>
<ls>

カレントディレクトリを表示します。

<compare 比較式>
<comparef 比較式>

比較式を評価。
compareは整数、comparefは不動小数点数の数式の比較を行う。

<erase 節名>
<retract 節名>

節名に該当するプログラム、変数、オブジェクトをすべて削除します。

<exit>

実行を途中で中断する。

<retractpred ヘッド>

ヘッドに該当するプログラムをすべて削除します。

<f 変数 述語リスト>
<func 変数 述語リスト>

述語リストの中に含まれている関数述語を実行評価して
値を置き換えます。すべてを実行後に結果を変数に設定
します。

<findall 述語...>

述語を実行して、すべての解を求めます。

<for (変数 実行回数) 述語...>
<for (変数 初期値 最終値) 述語...>

指定された回数、引数の述語を実行します。
変数には、順に数が設定されます。実行回数が指定された
場合は0からの値が、初期値が指定された場合はその値から
実行回数または最終値を超えない値まで、１ずつ増加させ
ます。
述語は１ターンの実行後にすべての変数のバインド
がクリアされて最初から実行されます。

<foreach (変数 リスト) 述語...>
<map (変数 リスト) 述語...>

リストの要素ごとに引数の述語を実行します。
変数には、リストの値が順に設定されます。
述語は１ターンの実行後に実行中の変数のバインド
がクリアされて最初から実行されます。

<help>
<help 項目名>

ヘルプマニュアルを表示する。
引数なしで実行すると、組み込み述語とsysモジュールのライブラリ
の述語の一覧を表示する。

<include ファイル名>

ファイル名のライブラリからモジュールを読み込みます。

<let 変数 = 数式>
<letf 変数 = 数式>
<letc 変数 = 数式>

数式を計算します。
左辺が変数の場合は、計算結果を代入します。
左辺が数値の場合は、計算結果と等しいか判定します。
letは、整数の計算を行い、letfは浮動小数点数の計算を
行います。
letcは複素数を計算します。

<list [変数]>

プログラムリストを表示します。

<load ファイル名>

ファイル名のプログラムを読み込みます。

〈loop 述語...〉

述語実行の繰り返しを示します。内部で呼び出された
述語が失敗するとループを抜けます。
述語は１ターンの実行後に実行中の変数のバインド
がクリアされて最初から実行されます。
述語の実行結果に関わらずに必ずTrueで成功する述語です。
"{述語...}"とも書けます。
EBNF記法の{}として構文解析で使います。

〈module 変数〉

現在使用しているモジュールを変数に設定します。

〈new〉

プログラムをすべてクリアします。

〈not 述語...〉

述語実行がtrueの場合はfalseを返します。
falseの場合はtrueを返します。
unknownの場合はunknownを返します。

〈obj オブジェクト名 述語...〉
〈unify モジュール名 述語...〉
指定されたオブジェクトまたはモジュールを使って、述語を
実行します。
オブジェクトとモジュールは、同等のものとして扱われるため
この２つは同じものです。
省略形として"::オブジェクト 〈述語...〉"と記述できます。

〈or 述語 述語 述語 ...〉

引数の述語を先頭から実行し、trueになったところで
処理を打ち切りtrueを返します。
引数の述語を先頭から実行し、falseになったところで
処理を打ち切りfalseを返します。
すべての述語がunknownの場合は、unknownを返します。
EBNF記法の　|　として構文解析で使います。

〈print リスト〉

リストを出力した後、改行します。

〈writenl リスト〉と同じである。

〈printf リスト〉

リストを出力します。
printとの違いはリストの要素の間に空白が入らないことと
自動的には改行しないことです。

〈% _ フォーマット 引数〉
printfのためにフォーマットの整形をします。

〈¥ _ 文字〉
　　エスケープ文字

〈printlist リスト〉

　　括弧を外して、リストを出力します。

〈printlistnl リスト〉

　　括弧を外して、要素毎に改行しながら、リストを出力します。

〈pwd [変数]〉

　　カレントディレクトリを変数に設定します。

〈quit〉

　　プログラムの実行を中止して終了します。

〈quote 述語〉

　　引数の関数述語の評価を抑止します。

〈rpn 変数 逆ポーランド式〉
〈rpnf 変数 逆ポーランド式〉
〈rpnc 変数 逆ポーランド式〉

　　逆ポーランド式を計算して、変数に結果を設定します。
　　rpnは、整数の計算を行い、rpnfは浮動小数点数の計算を
　　行います。
　　rpncは複素数の計算を行います。

〈save ファイル名〉

　　ファイル名のプログラムを書き込みます。

〈self 変数〉

　　変数に自身のオブジェクト名を設定する。

〈super 変数〉

　　変数に継承しているオブジェクト名のリストを設定する。

〈timeout 時間 述語〉

　　設定した時間以内に、述語の実行が終了しない場合は
　　処理を打ち切り、unknownを返します。
　　設定時間はマイクロ秒単位です。

〈troff〉

　　デバッグトレースをオフにします。

〈tron〉

デバッグトレースをオンにします。

<true>
<false>
<unknown>

true, false, unknownを返す

<warn　リスト>

リストをエラー出力に出力します。
出力後に改行します。

<x 述語...>

通常は何も行わない。
バックトラックしたときに引数の述語を実行する。

<!>

カット演算子

EBNF 構文解析

<TOKEN 変数 述語...>

入力の構文解析述語実行後に、得られたtokenを変数に
設定します。

<SKIPSPACE>

入力のスペースをスキップします。

<C [変数]>

入力を一文字変数に設定します。

<N [変数]>

入力が数字であった場合は、変数に設定します。
違う場合はunknownを返します。

<A [変数]>

入力がASCII文字であった場合は、変数に設定します。
違う場合はunknownを返します。

<AN [変数]>

入力がASCII文字か数字であった場合は、変数に
設定します。
違う場合はunknownを返します。

<^>

行の先頭とマッチする。

<$>
　　行の最後とマッチする。

<* [変数]>
　　任意の文字列とマッチする。

<CR>

　　入力がCR改行であった場合には、trueを返します。
　　違う場合はunknownを返します。

<CNTL [変数]>

　　入力がCNTL文字であった場合には、変数に設定します。
　　違う場合はunknownを返します。

<EOF>

　　入力がEOF(End Of File)である場合はtrueを返します。
　　違う場合はunknownを返します。

<SPACE>

　　入力がスペースである場合はtrueを返します。
　　違う場合はunknownを返します。

<PUNCT>

　　アルファベット、数字以外の文字である場合はtrue
　　を返します。

<WORD [変数]>

　　任意の文字列で、アルファベット、数字、"_"以外
　　の文字列の場合はunknownを返します。

<NUM [変数]>

　　入力の整数を変換して、変数に設定します。

<FNUM [変数]>

　　入力の浮動小数点数を変換して、変数に設定します。

<ID [変数]>

　　入力の文字列（先頭はアルファベット、それ以外
　　は数字も可)、合致すれば変数に設定します。

<RANGE 変数 文字１ 文字２>
<NONRANGE 変数 文字１ 文字２>

　　文字１と文字２の範囲に含まれるならばtrue
　　となります。

<GETTOKEN 変数>

直前の構文解析の結果であるトークンを変数に設定
します。

<SKIP 文字列>
文字列までの構文解析を行わずにスキップします。

<SKIPCR>
改行までの構文解析を行わずにスキップします。

<newObj 名前>

新しい名前のオブジェクトを生成する。

<delObj 名前>

名前のオブジェクトを削除する。

<method NAME>

The method of a new name is generated

<methoda NAME>

The method of a new name is generated from a head.

<methodz NAME>

The method of a new name is generated at the end.

<delmethod NAME>

The method of the specified name is deleted.

<delmethoda NAME>

The method of the specified name is deleted from a head.

<delmethodz NAME>

The method of the specified name is deleted from an end.

<setVar 変数名 値>

グローバル変数に値を設定します。
グローバル変数は、述語として以下の形式で登録
されます。
<変数名 値>;

<setArray 変数名 値 インデックス>

グローバル変数配列に値を設定します。
グローバル変数は、述語として以下の形式で登録
されます。
<変数名 値 インデックス>;

<delVar 変数名>

変数名に合致する変数が削除されます。

〈delArray 変数名 INDEX〉

変数名に合致する配列変数が削除されます。

〈変数名 値 インデックス〉;

多次元の配列を仕様する場合は、インデックスをリストとして指定します。

〈変数名 値 (インデックス1 インデックス2 ...)〉

インデックスは数以外にもいかなる種類や形式でも良いです。

以上。