

実験的仕様の関数

Risa/Asir 実験的仕様関数説明書
1.0 版
2006 年 3 月

by Risa/Asir committers

1 実験的仕様の関数

1.1 実験的仕様の関数説明書について

この説明書では Asir に導入された実験的仕様の関数について説明する。正式な関数として導入されたものの記述は Risa/Asir マニュアルに移動される。ChangeLog の項目は www.openxm.org の cvsweb でソースコードを読む時の助けになる情報が書かれている。

1.2 実験的仕様の関数

1.2.1 quotetotex, quotetotex_env

`quotetotex(q)`

:: q を latex 形式で表現した文字列に変換する。

`quotetotex_env(key, value)`

:: `quotetotex` の動作を制御するパラメータを変更する。

`quotetotex_env()`

:: `quotetotex` の動作を制御するパラメータの現在値を戻す。

`quotetotex_env(0)`

:: `quotetotex` の動作を制御するパラメータをデフォルト値に戻す。

return 文字列 (`quotetotex`) または リストまたはオブジェクト (`quotetotex_env`)

q quote

key 文字列

value オブジェクト

- `quotetotex` は q を latex 形式で表現した文字列に変換する。
- 以下 `quotetotex_env` のパラメータの意味を説明する。
- `conv_rule`: 3 ビットを用いて変換ルールを指定する。0 ビット目は `symbol_table` による変換を行うか, 1 ビット目は添字変換を行うか, 2 ビット目は d から始まる変数名を微分作用素とみなして処理するか, を意味する。たとえば `conv_rule` として 3 を指定すると, 0 ビット目, 1 ビット目が 1 となるので `symbol_table` による変換を行い, 添字変換をおこなう。添字変換は数字と英字の境目および $-$ 記号を区切りとする。 `symbol_table` による変換が最初に適用される。 `alpha`, `beta`, 等は自動的にギリシャ文字に変換するテーブルは内蔵済み。
- `dp_vars_prefix`: 分散表現多項式は x_0, x_1, \dots の多項式として latex 形式に変換されるがこの x の部分を変更する。
- `dp_vars_origin`: インデックスの始まりの値を指定する。デフォルトは 0。
- `dp_vars_hweyl`: 分散表現多項式をワイル代数の元とみなして latex 形式に変換する。偶数個変数があるときは最初の半分を x_0, x_1, \dots に後半の半分を $\partial_0, \partial_1, \dots$ に変換する。奇数個の場合は最後の変数が同時化変数として h で表示される。
- `dp_dvars_prefix`: `dp_vars_hweyl` が 1 の時に後半部分の `prefix` を指定する。デフォルトは ∂
- `dp_dvars_origin`: `dp_vars_hweyl` が 1 の時のインデックスの始まりの値。

- conv_func: ユーザ定義の変換関数をよぶ.

-

```
[3] quotetotex(quote(1/(x+1)));
\frac{ 1}{ { ( {x}+ 1)}}
[4] quotetotex(objtoquote(diff(x^x,x)));
{x}^{{ {x}- 1} {x}+ \log( {x}) {x}^{{ {x}}}}
[5] quotetotex_env("conv_rule",3);
[6] quotetotex(objtoquote( (alpha2beta+x_i_j)^2));
{\alpha}_{2,\beta}^{{ 2} + 2 {x}_{i,j} {\alpha}_{2,\beta}+ {x}_{i,j}^{{ 2}}
```

参照 Section 1.2.2 [objtoquote], page 2 print_tex_form(contrib)

ChangeLog

- この関数は 2004 年 2 月末から 3 月にかけて asir を knoppix 版 texmacs に対応させるために書かれた. Asir-contrib の print_tex_form がその原型であり, それを効率化した出力形式を改善した. OpenXM/src/kxx/ox_texmacs.c, OpenXM/src/texmacs も参照.
- OpenXM/src/asir-contrib/packages/src/noro_print.rr 1.1–1.8, noro_print_default.rr 1.1–1.3 も参照.
- 変更を受けたファイルは OpenXM_contrib2/asir2000 の下の次のファイル. builtin/strobject.c 1.14–1.43, include/ca.h 1.46, io/cexpr.c 1.18, io/pexpr.c 1.32, io.sexpr.c 1.29, parse/arith.c 1.12, parse/parse.h 1.28–1.29, parse/quote.c 1.7–1.8, 1.12.
- knoppix/math は 福岡大学の濱田さんが中心となり開発されている.
- dp_dvars_prefix, *_origin は builtin/strobject.c 1.46 で導入された.
- Todo: quotetoterminalform (分散表現多項式の見易い出力).

1.2.2 objtoquote

objtoquote(ob)

:: オブジェクトと quote 型のデータに変換する.

return quote

ob オブジェクト

- objtoquote(ob) は, ob を quote 型のデータに変換する.

```
[1150] quotetolist(quote(1+2));
[b_op,+, [internal,1], [internal,2]]
[1151] quotetolist(objtoquote(1+x));
[b_op,+, [internal,x], [internal,1]]3
```

参照 \langle undefined \rangle [quotetotex], page \langle undefined \rangle \langle undefined \rangle [quotetolist], page \langle undefined \rangle

ChangeLog

- この関数は quotetotex の前処理をするために書かれた.
- asir-contrib の関数 quote_to_quote も参照.
- OpenXM_contrib2/asir2000/builtin/print.c 1.16.

1.2.3 copyright

copyright()

:: Risa/Asir の copyright 表示を文字列として戻す.

return 文字列

- Risa/Asir の copyright 表示を文字列として戻す.

```
[1150] copyright();
```

```
This is Risa/Asir, Version 20040312 (Kobe Distribution).
```

```
Copyright (C) 1994-2000, all rights reserved, FUJITSU LABORATORIES LIMITED.
```

```
Copyright 2000-2003, Risa/Asir committers, http://www.openxm.org/.
```

```
GC 6.2(alpha6) copyright 1988-2003, H-J. Boehm, A. J. Demers, Xerox, SGI, HP.
```

```
PARI 2.0.17, copyright 1989-1999, C. Batut, K. Belabas, D. Bernardi,
```

```
H. Cohen and M. Olivier.
```

ChangeLog

- この関数は texmacs 用に書かれた (2004-03).
- OpenXM_contrib2/asir2000 の下の以下のファイルをみよ. builtin/miscf.c 1.21, parse/glob.c 1.47.

1.2.4 string_to_tb, tb_to_string, write_to_tb

string_to_tb(s)

tb_to_string(tb)

write_to_tb(s,tb)

:: 文字列可変長配列型 (text buffer) のデータの処理

return 文字列可変長配列型 (string_to_tb), 文字列型 (tb_to_string)

s 文字列

tb 文字列可変長配列型

- string_to_tb(s) は, 文字列 s をはじめの要素とする文字列可変長配列型オブジェクトを生成する.
- tb_to_string(tb) は, 文字列可変長配列型オブジェクト tb から通常の文字列オブジェクトを生成する.
- write_to_tb(s,tb) は, 文字列 s を文字列可変長配列型オブジェクト tb へ書き出す.
- SS を文字列変数とするとき, SS += "文字列" で SS へ文字列を書き足していくことができるが, 無駄なメモリを大量に消費する. 代りに関数 write_to_tb を用いるべきである. 文字列可変長配列型オブジェクトは文字列の可変長の配列でありメモリ管理に優しいデータ構造である.

```
[219] T=string_to_tb("");
```

```
[220] write_to_tb("Hello",T);
```

```
0
```

```
[221] write_to_tb(" world!",T);
```

```
0
```

```
[222] tb_to_string(T);
```

```
Hello world!
```

ChangeLog

- この関数は 2004-3 に `print_tex_form` を効率化するために書かれた.
- `OpenXM_contrib2/asir2000` の下の以下のファイルを見よ. `io/ox_asir.c` 1.52, `builtin/strobject.c` 1.12–1.13, 1.16, `engine/str.c` 1.5, `parse/quote.c` 1.9.
- `rtostr` が `text buffer` 型のデータに関しておそかった. 速度の改善は `asir2000/io/pexpr_body.c` 1.2, `asir2000/parse/lex.c` 1.32.

1.2.5 dp_gr_main

`dp_gr_main(f | v=vv, order=oo, homo=n, matrix=m, block=b, sugarweight=sw)`
 :: `dp-gr-main` の新しいインタフェース.

`return` リスト (グレブナ基底. 再帰表現多項式か分散表現多項式のリスト)

`f` リスト (入力多項式系. 再帰表現多項式か分散表現多項式のリスト)

`vv` リスト (変数のリスト)

`oo` リスト (順序をあらわすリスト)

`n` 0 か 1 (homogenization をするか)

`m` 順序を `matrix` で表現する場合 (cf. `dp_ord`).

`b` ???

`sw` Sugar strategy を適用するときの weight vector. 全ての要素は非負.

- `dp_gr_main(f)` は, `f` のグレブナ基底を計算する. グレブナ基底は順序を変えるとその形が変わる. `asir` ではいままで順序の指定方法が系統だっていなかった. `dp-gr-main` の新しいインタフェースでは順序のある文法に従い指定する.
- 順序 `order` は次の文法で定義する. `{, }` は 0 回以上の繰り返しを意味する.

```

order      : '[' orderElement ',' orderElement ']'
orderElement : weightVec | builtinOrder
weightVec   : '[' weightElement ',' weightElement ']'
builtinOrder : '[' orderName ',' setOfVariables ']'
weightElement : NUMBER | setOfVariables ',' NUMBER
setOfVariables: V | range(V,V)
orderName    : @grlex | @glex | @lex

```

ここで `V` は 変数名, `NUMBER` は整数をあわらす. 例 1: `v=[x,y,z,u,v]`, `order=[[x,10,y,5,z,1],[@grlex,range(x,v)]]` は `x,y,z` がそれぞれ weight 10, 5, 1 をもつ順序で比較したあと, `[x,y,z,u,v]` についての graded reverse lexicographic order を tie-breaker として用いることを意味する. 参考書: B.Sturmfels: *Gr\obner Bases and Convex Polytopes* (1995). M.Saito, B.Sturmfels, N.Takayama: *Gr\obner Deformations of Hypergeometric Differential Equations* (2000).

- 順序要素 (`orderElement`) の指定方法は (1) 変数名または `range` で指定された変数の集合と重みの値の繰り返し (2) 重みの値を変数リストの順番に並べる方法 (3) 変数名または `range` で指定された変数の集合と順序名の組の三通りの基礎的方法がある. 似た指定方法が `Macaulay`, `Singular`, `CoCoA`, `Kan/sm1` 等の環論システムで使用されていた. `Risa/Asir` の指定方法はこれらのシステムの指定方法を参考にさらに改良を加えたもので柔軟性が高い.
- `order` の tie-breaker は `grlex` がデフォルト.

- 分散表現多項式を引数としたときは結果も分散表現多項式として戻る. order 指定にもちいるデフォルトの変数名はこのとき x_0, x_1, x_2, \dots となる.
- オプションの値は option-list キーワードを用いてリストで与えてもよい. 下の例を参照.

```
[218] load("cyclic");
[219] V=vars(cyclic(4));
[c0,c1,c2,c3]
[220] dp_gr_main(cyclic(4) | v=V, order=[[c0,10,c1,1],[c2,5],[@grlex,range(c0,c3)]])
[ 10 1 0 0 ]
[ 0 0 5 0 ]
[ R R R R ]
[(-c3^6+c3^2)*c2^2+c3^4-1,c3^2*c2^3+c3^3*c2^2-c2-c3,
 (c3^4-1)*c1+c3^5-c3,(c2-c3)*c1+c3^4*c2^2+c3*c2-2*c3^2,-c1^2-2*c3*c1-c3^2,
 c0+c1+c2+c3]

[1151] F=map(dp_ptod,katsura(4), vars(katsura(4)));
[(1)*<<1,0,0,0>>+(2)*<<0,1,0,0,0>>+ ... ]
[1152] dp_gr_main(F | order=[[range(x0,x3),1]]);
[ 1 1 1 1 0 ]
[ R R R R R ]
[(47774098944)*<<0,0,0,0,13>>+ ... ]

[1153] Opt=[["v",[x,y]], ["order",[x,5,y,1]]];
[[v,[x,y]], [order,[x,5,y,1]]]
[1154] dp_gr_main([x^2+y^2-1,x*y-1] | option_list=Opt);
[ 5 1 ]
[ R R ]
[-y^4+y^2-1,x+y^3-y]
```

ChangeLog

- この関数は 2003-12 から 2004-2 の始めに大きな修正が行われた.
- setOfVariablesの表現のために range オブジェクトが導入された.
- グレブナ基底は順序を変えるとその形が変わる. asir ではいままで順序の指定方法が系統だっていなかった. dp_gr_main の新しいインタフェースでは順序のある文法に従い指定する.
- OpenXM_contrib2/asir2000 の下の次の各ファイルが修正をうけた. builtin/gr.c 1.56-1.57, builtin/dp-suppl.c 1.27-1.31 (create_composite_order_spec), builtin/dp.c 1.46-1.48 (parse_gr_option), engine/Fgfs.c 1.20, engine/dist.c 1.27-1.28 engine/nd.c 1.89, include/ca.h 1.42-1.43, io/pexpr.c 1.28, io/sexpr.c 1.26, parse/arith.c 1.11, parse/glob.c 1.44-1.45, parse/lex.c 1.29, parse/parse.h 1.23-1.26
- Todo: return キーワードで戻り値のデータを quote のリストにできるように. attribute, ring 構造体.

1.2.6 asir-port.sh, asir-install.sh

asir-install.sh

asir-port.sh

:: これは asir の内部コマンドではない. asir をネットワークからダウンロードかつ実行するシェルスクリプト

- asir-port.sh は knoppix 専用である. このコマンドは asir のバイナリおよび FLL で配布できない部分を ftp.math.kobe-u.ac.jp よりダウンロードして/home/knoppix/.asir-tmpヘセーブして, 実行する. .asirrc および .TeXmacs/plugins/ox/progs/init-ox.scm もダウンロードする.
- asir-install.sh は Debian GNU Linux / openxm-binary*.deb 専用である. asir-install.sh は asir をダウンロードして /usr/local/OpenXM/bin および /usr/local/OpenXM/lib/asir ヘインストールする.

ChangeLog

- これらのシェルスクリプトは knoppix/math のために 2004/2, 3 月に書かれた.
- knoppix/math は福岡大学の濱田さんが中心となり開発されている.
- OpenXM/misc/packages/Linux/Debian の下の全てのファイル (2004-2-22 から 2004-3の末まで). (~taka/this03/misc-2003/A3/knoppix-03-05 (プライベートファイル) も見よ.) OpenXM/src/asir-port の下の次の各ファイル. Makefile 1.1-1.8, asir-install.sh 1.1-1.2, asir-port.sh 1.1-1.6.

1.2.7 get_struct_name, get_element_names, get_element_at, put_element_at

get_struct_name(s)

get_element_names(s)

get_element_at(s, key)

put_element_at(s, key, obj)

:: 構造体 s に対する操作

return 文字列 (get_struct_name), 文字列のリスト (get_element_names), オブジェクト (get_element_at), オブジェクト (put_element_at)

s 構造体

key 文字列

obj オブジェクト

- get_struct_name(s) は, 構造体 s の名前を戻す.
- get_element_names(s) は, 構造体のメンバーの名前のリストを戻す.
- get_element_at(s, key) は構造体 s のメンバー key の値を戻す.
- put_element_at(s, key, obj) は構造体 s のメンバー key の値を obj に設定する.

```
[219] struct point x, y, color;
[220] P = newstruct(point);
0,0,0
[221] P->x = 10$ P->y=5$ P->color="red"$
[222] get_element_names(P);
[x,y,color]
```



```

[223] put_element_at(P,"color","blue");
blue
[224] P->color;
bule

```

参照 [〈undefined〉 \[newstruct\]](#), [page 〈undefined〉](#), [〈undefined〉 \[struct\]](#), [page 〈undefined〉](#)

ChangeLog

- 構造体の定義を知らずに構造体を扱うユーザ関数を書くときに便利. asir-contrib の `noru_print.rr` を見よ.
- `OpenXM_contrib2/asir2000/builtin/compobj.c` 1.8.

1.2.8 dp_weyl_gr_main

`dp_weyl_gr_main(f | v=vv, order=oo, homo=n, matrix=m, block=b, sugarweight=sw)`

:: `dp_weyl_gr_main` の新しいインタフェース. `dp_gr_main` と同じ形式である.

return リスト (グレブナ基底. 再帰表現多項式か分散表現多項式のリスト)

f リスト (入力多項式系. 再帰表現多項式か分散表現多項式のリスト)

vv リスト (変数のリスト)

oo リスト (順序をあらわすリスト)

n 0 か 1 (homogenization をするか). [テストまだ]

m 順序を `matrix` で表現する場合 (cf. `dp_ord`). [テストまだ]

b ???

sw Sugar strategy を適用するときの weight vector. 全ての要素は非負. [テストまだ]

- `dp_weyl_gr_main(f)` は, f のグレブナ基底を計算する. グレブナ基底は順序を変えるとその形が変わる. asir ではいままで順序の指定方法が系統だっていなかった. `dp_weyl_gr_main` の新しいインタフェースでは順序のある文法に従い指定する. 指定方法については `dp_gr_main` のマニュアルを参照.
- 分散表現多項式の各モノミアルの長さが偶数のときはワイル代数 $K[x_1, \dots, x_n, d_1, \dots, d_n]$ で計算がおこなわれる. ワイル代数では x_i と d_i は非可換な掛け算規則 $d_i x_i = x_i d_i + 1$ をみたし, x_i と x_j や d_i と d_j は可換である. また i と j が異なる場合は x_i と d_j も可換である.
- 分散表現多項式の各モノミアルの長さが奇数のときは同次化ワイル代数 $K[x_1, \dots, x_n, d_1, \dots, d_n, h]$ で計算がおこなわれる. 同次化ワイル代数では x_i と d_i は非可換な掛け算規則 $d_i x_i = x_i d_i + h^2$ をみたし, h は任意の元と可換, その他の変数もワイル代数と同様な可換性の規則をみたす. 詳しくは `dp_gr_main` で参照した Saito, Sturmfels, Takayama の教科書をみよ.

```

[1220] F=sm1.gkz([ [1,1,1,1],[0,1,3,4]], [0,0]); /* Command in asir-contrib */
[[x4*dx4+x3*dx3+x2*dx2+x1*dx1,4*x4*dx4+3*x3*dx3+x2*dx2,-dx1*dx4+dx2*dx3,-dx2^2*dx4+
[1221] V=[x1,x2,x3,x4,dx1,dx2,dx3,dx4]$
[1222] dp_weyl_gr_main(F[0] | v=V, order=[[dx1,1,dx2,1,dx3,1,dx4,1]]);
...

```

```

[1238] FF=map(dp_ptod,F[0],V);
(1)*<<1,0,0,0,1,0,0,0>>+(1)*<<0,1,0,0,0,1,0,0>>+(1)*<<0,0,1,0,0,0,1,0>>+(1)*<<0,0,0,1,0,0,0,1>>+(1)*<<0,0,0,0,1,0,0,0>>+(1)*<<0,0,0,0,0,1,0,0>>+(1)*<<0,0,0,0,0,0,1,0>>+(1)*<<0,0,0,0,0,0,0,1>>+(1)*<<0,0,0,0,0,0,0,0>>

[1244] FF=map(dp_ptod,F[0],V);
(1)*<<1,0,0,0,1,0,0,0>>+(1)*<<0,1,0,0,0,1,0,0>>+(1)*<<0,0,1,0,0,0,1,0>>+(1)*<<0,0,0,1,0,0,0,1>>+(1)*<<0,0,0,0,1,0,0,0>>+(1)*<<0,0,0,0,0,1,0,0>>+(1)*<<0,0,0,0,0,0,1,0>>+(1)*<<0,0,0,0,0,0,0,1>>+(1)*<<0,0,0,0,0,0,0,0>>

dp_weyl_gr_main(FF | v=V, order=[[0,0,0,0,1,1,1,1]]);

[1246] dp_weyl_gr_main(FF | v=V, order=[[dx1,1,dx2,1,dx3,1,dx4,1]]);
[ 0 0 0 0 1 1 1 1 ]
[ R R R R R R R R ]
...

```

参照 Section 1.2.5 [dp-gr-main], page 4

ChangeLog

- dp-gr-main のインタフェースが dp_weyl_gr_main へも導入された.
- OpenXM_contrib2/asir2000 の下の次の各ファイルが修正をうけた. builtin/dp-supp.c 1.32–1.33 builtin/dp.c 1.49–1.50

1.2.9 dp_initial_term

dp_initial_term(*f* | *v*=*vv*, *order*=*oo*)
 :: dp_initial_term は与えられた *weight* に対する先頭項の和を戻す.

return 分散表現多項式または分散表現多項式のリスト.

f 分散表現多項式か分散表現多項式のリスト.

vv リスト (変数のリスト)

oo リスト (順序をあらわすリスト)

- dp_initial_term は与えられた *weight* *w* に対する先頭項の和を戻す. これは多くの教科書で $\text{in}_w(f)$ と書かれている.
- 順序を表すリストは dp-gr-main で定義した文法に従う. このリストの先頭が *weight vector* で無い場合はエラーとなる. たとえば *order*=[[@lex,...]] はエラーとなる.
- 結果は与えられた順序に関してソートされてるわけではない.

```

[1220] F=<<2,0,0>>+<<1,1,0>>+<<0,0,1>>;
(1)*<<2,0,0>>+(1)*<<1,1,0>>+(1)*<<0,0,1>>
[1220] dp_initial_term(F | order=[[1,1,1]]);
[ 1 1 1 ]
[ R R R ]
(1)*<<2,0,0>>+(1)*<<1,1,0>>
[1221] dp_initial_term(F | v=[x,y,z], order=[[x,1]]);
[ 1 0 0 ]
[ R R R ]
(1)*<<2,0,0>>

```

参照 Section 1.2.5 [dp-gr-main], page 4, Section 1.2.8 [dp_weyl_gr-main], page 7, Section 1.2.10 [dp-order], page 9, (undefined) [dp-hm], page (undefined)

ChangeLog

- OpenXM_contrib2/asir2000 の下の次の各ファイルが修正を受けた. builtin/dp-supp.c 1.32 builtin/dp.c 1.49

1.2.10 dp_order

`dp_order(f | v=vv, order=oo)`

:: `dp_order` は与えられた `weight` に対する次数の最大値を戻す.

return 数か数のリスト

f 分散表現多項式か分散表現多項式のリスト.

vv リスト (変数のリスト)

oo リスト (順序をあらわすリスト)

- 順序を表すリストは `dp_gr_main` で定義した文法に従う. このリストの先頭が `weight vector` で無い場合はエラーとなる. たとえば `order=[[lex,...]]` はエラーとなる.
- `dp_order` は与えられた `weight w` に対する次数の最大値を戻す. これを $\text{ord}_w(f)$ と書く論文や教科書もある.
- 引数がリストの場合各要素の次数が計算される.

```
[1220] F=<<2,0,0>>+<<1,1,0>>+<<0,0,1>>;
(1)*<<2,0,0>>+(1)*<<1,1,0>>+(1)*<<0,0,1>>
[1222] dp_order(F | order=[[1,1,1]]);
[ 1 1 1 ]
[ R R R ]
2
[1223] dp_order(F | v=[x,y,z], order=[[x,1]]);
[ 1 0 0 ]
[ R R R ]
```

参照 Section 1.2.5 [`dp_gr_main`], page 4, Section 1.2.8 [`dp-weyl-gr_main`], page 7, Section 1.2.9 [`dp_initial_term`], page 8, (undefined) [`dp_hm`], page (undefined)

ChangeLog

- OpenXM_contrib2/asir2000 の下の次の各ファイルが修正を受けた. builtin/dp-supp.c 1.32 builtin/dp.c 1.49

1.2.11 mapat

`mapat(fname,pos[,arg0, arg1, ...])`

:: `pos` に対する `map` 関数

return オブジェクト

pos 整数

arg0, arg1, arg2, ...

オブジェクト

- `map` 関数は 0 番目の引数に対してしか動作しないが, `mapat` 関数は指定した番号の引数に対して `map` 関数を実行する.
- `mapat(fname,0,A0,A1,...)` は `map(fname,A0,A1,...)` に等価である.

- 次の副作用がある. まだ書いてない.

```
[219] mapat(deg,1,x^2+y^3+x+y,[x,y]);
[2,3]
[220] mapat(subst,1,x+y+z,[x,y,z],2);
[y+z+2,x+z+2,x+y+2]
```

参照 [\(undefined\) \[map\]](#), page [\(undefined\)](#)

ChangeLog

- この関数は 2004-6-22 にコミットされた. 変更をうけたソースコードは builtin/pf.c, subst.c である.

1.2.12 list

```
list([arg0, arg1, ...])
:: list を生成する.
```

return リスト

arg0, arg1, arg2, ...
オブジェクト

- arg0, arg1, ... を要素とするリストを生成する.

```
[219] list(1,2,3);
[1,2,3]
[220] list(1,2,[3,4]);
[1,2,[3,4]]
```

参照 [\(undefined\) \[cons\]](#), page [\(undefined\)](#)

ChangeLog

- この関数は 2004-6-22 にコミットされた. 変更をうけたソースコードは builtin/list.c である.

1.2.13 set_print_function

```
set_print_function([fname])
:: 画面表示用の関数を登録
```

return 整数

fname 文字列

- set_print_function は fname(F) を通常の画面表示関数の代わりによぶ. 引数がない場合は画面表示関数をデフォルトへ戻す. Asir-contrib はこの関数を用いて出力関数を Asir-contrib 用に変更している.

```
[219] def my_output(F)
      print("Out: ",0); print(rtostr(F));
```

```
[220] set_print_function("my_output");
Out: 0
[221] 1+2;
Out: 3
```

参照 `<undefined>` [rtostr], page `<undefined>`

ChangeLog

- この関数は 2001-9-4 に asir-contrib のために導入された. 変更をうけたソースコードは builtin/print.c 1.11 である.

1.2.14 small_jacobi

`small_jacobi(a,m)`
:: Jacobi 記号の計算

return 整数

arg1, arg2 整数

- m が素数のときは Legendre 記号とよばれ, $x^2 = a \bmod m$ に解があるとき 1, 解がないとき -1 をもどす.
- Jacobi 記号は Legendre 記号の積で定義される (初等整数論の本参照).
- この関数は machine int の範囲で jacobi 記号を計算する.

```
[1286] small_jacobi(2,3);
-1
[1287] small_jacobi(2,7);
1
```

参照 <http://members.jcom.home.ne.jp/yokolabo/asirlib/> も見てね.

ChangeLog

- この関数の由来は不明.

1.2.15 flatten_quote

`flatten_quote(q,op)`
:: quote の括弧をとりさる.

return Quote

q Quote

op 演算子を表す文字列.

- Quote 型のデータは木構造をしている (quotetolist 参照). `quote_flatten()` は, q の中にあられる演算子 op の子供ノードを平等にする. つまり演算子 op に関する括弧づけがあった場合それをすべてとりさる. たとえば $(1+2)+(3+4)$ という表現を $1+2+3+4$ に変換する.
- 現在の実装では n -ary の演算子は定義されていないので, $1+2+3$ は実は $1+(2+3)$ と表現されている. つまり $+$ 演算子は右結合的である.
- $R=0$; for ($I=0$; $I<N$; $I++$) $R = R + P[I]$; なる足し算を繰り返すと, $+$ は左結合的になる. 右結合的に変換するには `flatten_quote` を呼ぶ.
- 名前は `quote_flatten` でなく `flatten_quote` である.

```
[1288] flatten_quote(quote((1+2)+(3+4*(x+3))), "+");
quote(1+2+3+4*(x+3))
[1289] flatten_quote(quote( (x*y)*(p*3)-(x*y)*z ), "*");
quote(x*y*p*3-x*y*z)
```

```
[1290] quotetolist(quote(1+2+3));
[b_op,+, [b_op,+, [internal,1], [internal,2]], [internal,3]]
```

参照 `<undefined> [quotetolist]`, page `<undefined>`, `<undefined> [print_tex_form]`,
page `<undefined>(contrib)`

ChangeLog

- この関数は 2004-7-7 から 2004-7-8 にかけて `quote` に関する操作を研究するために実験的に書かれた。OpenXM/fb で蓄積された公式の不要な括弧をとりはずし, `tex` 形式に変換するのに応用。
- 変更をうけたソースコードは builtin/strojb.c 1.47, parse/eval.c 1.35, parse/parse.h 1.31, parse/quote.c 1.14–1.16.

1.2.16 assoc

`assoc(a,b)`

:: 連想リストをつくる

return List

a List

b List

- リスト *a*, *b* より `[[a[0],b[0]], [a[1],b[1]], ...]` なる新しいリストを生成する。

下の例では *A* に動物の名前が, *B* に足の本数が入っている。 `assoc(A,B)` で動物と足の本数をペアにしたリストを生成する。

```
[1192] A=["dog","cat","snake"];
[dog,cat,snake]
[1193] B=[4,4,0];
[4,4,0]
[1194] assoc(A,B);
[[dog,4],[cat,4],[snake,0]]
```

参照 `<undefined> [cons]`, page `<undefined>`, `<undefined> [append]`, page `<undefined>`

ChangeLog

- この関数は 2004-6-28 に書かれた。 変更をうけたソースコードは builtin/list.c 1.9 parse/eval.c 1.35, parse/parse.h 1.31, parse/quote.c 1.14–1.16.

1.2.17 sprintf

`sprintf(format[,args])`

:: C に似たプリント関数

return 文字列

format 文字列

args オブジェクト

- フォーマット文字列 *format* にしたがって *args* を文字列に変換する。
- フォーマット文字列の中に `%a` (any) が利用可能。 *args* の個数はフォーマット文字列の中の `%a` の個数に等しくすること。

```
[0] sprintf("%a: rat = %a",10,x^2-1);
10: rat = x^2-1
```

参照 [\(undefined\)](#) [rtostr], page [\(undefined\)](#)

ChangeLog

- この関数は 2004-7-13 にコミットされた. 変更をうけたソースコードは builtin/strobj (1.50) である.
- %a は Maple の sprintf の真似か.

1.2.18 quote_to_funargs, funargs_to_quote, remove_paren

quote_to_funargs(*q*)

:: quote を funarg 形式 (リスト) へ.

funargs_to_quote(*f*)

:: funarg 形式を quote へ.

get_function_name(*f*)

:: funarg 形式の op を文字列へ.

remove_paren(*q*)

:: 上の関数を用いて書かれた余分な括弧を取り去る simplifier (asir-contrib マニュアルへ: todo)

return quote(funargs_to_quote, remove_paren) か リスト (quote_to_funargs)

q quote

f リスト

- quote_to_funargs は quote 型のデータ (内部的には FNODE) を quote への復元可能な形でリストへ変換する. quotetolist は quote をリストへ変換するが, 一部の情報を捨てるためもとの quote の復元はできない.
- quote_to_funargs の戻り値は [fid, op, arg1, arg2, ...] なる形式をしている. ここで op は node の名前であり, 関数 get_function_name を用いて人間が読める形式で取りだせる. たとえば get_function_name(quote_to_funargs(quote(1+2))[1]) は "+" を戻す.
- 名前 get_function_name はそのうち変更されるだろう.
- 下の例で quote_to_funargs(FA[2]); [34,[b-op,+, [internal,x], [internal,1]]] となる. 34 は I_PAREN を意味する. 数と意味の対応表は OpenXM/src/asir-contrib/packages/src/noro_simplify.rr または OpenXM_contrib2/asir2000/parse/parse.h を見よ. 以下の fid が 0, 1, 2, ... に対応づけられている. LBOP, LCOP, LAND, LOR, LNOT, LCE, LPRESELF, LPOSTSELF, LFUNC, LFUNC_OPT, LIFUNC, LMAP, LRECMAP, LPFDERIV, LANS, LPVAR, LASSPVAR, LFORMULA, LLIST, LSTR, LNEWCOMP, LCAR, LCDR, LCAST, LINDEX, LEV, LTIMER, LGF2NGEN, LGFPNGEN, LGFSNGEN, LLOP, LOPT, LGETOPT, LPOINT, LPAREN, LMINUS, LNARYOP

次の例では (x+1)+(x+2) の括弧をはずして x+1+x+2 に変換している.

```
[0] ctrl("print_quote",1) $
```

```

[1] Q=quote((x+1)+(x+2));
[b_op,+, [u_op,(), [b_op,+, [internal,x], [internal,1]]],
      [u_op,(), [b_op,+, [internal,x], [internal,2]]]]

[2] FA=quote_to_funargs(Q);
[0,<...quoted...>,
  [u_op,(), [b_op,+, [internal,x], [internal,1]]],
  [u_op,(), [b_op,+, [internal,x], [internal,2]]]]

[3] FA2=quote_to_funargs(FA[2])[1];
[b_op,+, [internal,x], [internal,1]]

[4] FA3=quote_to_funargs(FA[3])[1];
[b_op,+, [internal,x], [internal,2]]

[5] funargs_to_quote([FA[0],FA[1],FA2,FA3]);
[b_op,+, [b_op,+, [internal,x], [internal,1]],
  [b_op,+, [internal,x], [internal,2]]]

```

次の例は OpenXM/asir-contrib 版の asir で実行.

```

[1287] load("noro_simplify.rr");
1
[1293] noro_simplify.remove_paren(quote( f(1-(x))));
quote(f(1-x))

```

funargs_to_quote を用いて既存の quote の子供を置き換えて新しい quote をつくり出せる.

```

[1184] R=quote_to_funargs(quote(a+(b+c)));
[0,<...quoted...>,<...quoted...>,<...quoted...>]
[1185] T=quote_to_funargs(quote(1+2));
[0,<...quoted...>,<...quoted...>,<...quoted...>]
[1186] funargs_to_quote([0,R[1],R[2],T[2]]);
quote(a+1)

```

参照 [\[undefined\]](#) [quotetolist], page [\[undefined\]](#)

ChangeLog

- これらの関数は 2004-7-8 から開発のはじまっている quote の simplification 関連の実験的関数である. 変更をうけたソースコードは多岐にわたるのでまだ書かない.
- 括弧を取り去る問題は OpenXM/fb が蓄えている公式を tex で綺麗に表示するのが動機の一つ.
- 2004-6-26 の計算代数セミナーにおいて, 中川さんが simplifier についていろいろ問題提起をした (計算代数セミナービデオ参照).
- parse/quote.c の `\tt strcut fid_spec fid_spec_tab[]` の部分に書いてある形式に funargs_to_quote は変換する.

1.2.19 set_secure_flag, set_secure_mode

set_secure_flag(fname,m)

set_secure_mode(m)

:: 関数の実行権限を設定する. (web サービス用)

return 整数

fname 文字列

m 整数

- `set_secure_flag`, `set_secure_mode` は `asir` を web サーバ等で公開するために加えられた関数. `set_secure_flag` で公開する関数を指定する. `secure_mode` が 1 の場合は `set_secure_flag` で指定された関数しか実行できない. 関数の実行途中では `secure_mode` が 0 となっているので, 任意の関数を実行できる. またエラーの時等は, `secure_mode` は 1 に自動的に復帰する. ただし `def` は実行できない. 公開する関数では, その処理中は任意の関数が実行できるので, `security` に十分注意した実装をする必要がある.
- `set_secure_flag` は, `fname` の `secure flag` を `m` にする. 公開する命令は 1 に設定する.
- `set_secure_mode(1)` で `secure_mode` が 1 となり, 公開された関数しか実行できなくなる. `quit` 等も実行できないので注意.
- `timer` の引数として `secure_flag` を設定していない関数を指定して実行してもエラーが表示されない. このときは, `ctrl("error_in_timer",1)` を実行しておく.

```
[1194] set_secure_flag("print_input_form_",1);
```

```
1
```

```
[1195] set_secure_flag("fctr",1);
```

```
1
```

```
[220] set_secure_mode(1);
```

```
1
```

```
[1197] fctr((x-1)^3);
```

```
[[1,1],[x-1,3]]
```

```
[1198] fctr(shell("ls"));
```

```
evalf : shell not permitted
```

```
return to toplevel
```

参照 `<undefined>` [`timer`], page `<undefined>`

ChangeLog

- `set_secure_flag`, `set_secure_mode` は `asir` を web サーバ等で公開するために加えられた関数. `sm1` の同様な関数 `RestrictedMode` で採用された方法を用いている. つまり, `set_secure_flag` で公開する関数を指定する. `secure_mode` が 1 の場合は `set_secure_flag` で指定された関数しか実行できない. `v` 関数の実行途中では `secure_mode` が 0 となっているので, 任意の関数を実行できる.
- この関数は 2004-10-27 から 2004-11-22 にかけて開発された.
- `cgi-asir.sm1`, `cgi-asir.sh` と組み合わせて `cgi` サービスを提供するために利用する. `cgi-asir.sh` では `CGI_ASIR_ALLOW` 環境変数で公開するコマンドを指定する.
- 1.24-1.25 `OpenXM_contrib2/asir2000/builtin/miscf.c`
- 1.36-1.38 `OpenXM_contrib2/asir2000/parse/eval.c`
- 1.6-1.7 `OpenXM_contrib2/asir2000/parse/function.c`
- 1.33 `OpenXM_contrib2/asir2000/parse/parse.h`

1.2.20 initialize_static_variable

`static` 変数の初期化の問題点. 初期化の時に `segmentation fault` がおきる.

- `static` 変数の取扱. 下の例を参照のこと.

```

if (1) {
  module abc;
  static A;
  A=1;
  endmodule;
} else { };

```

end\$

を t.rr とするとき,

```

[6] load("./t.rr");
1
internal error (SEGV)
となる.

```

t.rr を

```

if (1) {
  module abc;
  static A;
  localf initA;
  localf foo;
  def initA() {
    A=1;
  }
  initA();
  def foo() {
    return A;
  }
  endmodule;
} else { };

```

end\$

とすると正しく初期化される.

ChangeLog

- oxasir-win.rr の取扱で問題点として浮上. 2005.07.25.
- oxasir-win.rr の取扱で double quote の取り扱いに問題があったが, これは asir2000/io/ox_asir.c, 1.58, で問題点解決.
- ox_asir に計算を依頼する時は if (1) { ... } で囲む.

1.2.21 eval_quote

eval_quote(Q);

:: quote 型データ Q を asir のオブジェクトに変換する.

return オブジェクト

Q quote 型

- quote 型データ Q を asir のオブジェクトに変換する.

- 逆関数は objtoquote

```
ctrl("print_quote",2);
A=quote((x-1)^2+(x-1)+3);
  出力: (((x)-(1))^(2))+((x)-(1)))+(3)
eval_quote(A);
  出力: x^2-x+3
print_input_form(A);      /* asir-contrib */
  出力: quote((x-1)^2+(x-1)+3)
```

参照 Section 1.2.2 [objtoquote], page 2, [\(undefined\)](#) [quotetolist], page [\(undefined\)](#),
[\(undefined\)](#) [eval_string], page [\(undefined\)](#), Section 1.2.18 [quote_to_funargs],
page 13, [\(undefined\)](#) [funargs_to_quote], page [\(undefined\)](#)

ChangeLog

- —まだ書いてない.

1.2.22 nqt_match

`nqt_match(Expr, Pattern[, Mode])`
:: *Expr* が *Pattern* にマッチ (適合) すると 1 を戻す. しないと 0 を戻す.

return 整数

Expr quote 型

Pattern quote 型

Mode 整数

- *Expr* が *Pattern* にマッチ (適合) すると 1 を戻す. しないと 0 を戻す.
- 適合した場合, 副作用として, *Pattern* に含まれるプログラム変数 (大文字ではじまる変数) に適合した値が代入される.
- `nqt` は normalized quote の略であり `fnode` 標準形に変換してから適合検査をする. `fnode` 標準形については Section 1.2.24 [qt_normalize], page 18 を見よ.
- *Mode* により展開方法を指定し, その展開方法により得られた *Expr* の `fnode` 標準形と *Pattern* を比較する.

```
ctrl("print_quote",2);
A=quote((x-y)*(x+y));
nqt_match(A,quote(P*Q));
[P,Q]
  出力: [x-y, x+y]
nqt_match(A,quote(P*Q),1);
  マッチしない.
nqt_match(A,quote(P*Q),2);
  マッチしない.
qt_normalize(A,1);
  出力: ((x)^(2))+((x)*(y))+((-1)*((y)^(2)))+((-1)*(y)*(x))
qt_normalize(A,2);
  出力: ((x)*(x))+((x)*(y))+((-1)*(y)*(x))+((-1)*(y)*(y))
```

参照 Section 1.2.23 [nqt_match_rewrite], page 18, Section 1.2.30 [qt_rewrite], page 22

ChangeLog

- — まだ書いてない.

1.2.23 nqt_match_rewrite

`nqt_match_rewrite(Expr, Rule, Mode)`

:: *Expr* を *Rule* に従い書き換える.

return quote 型

Expr quote 型

Rule [*Pattern*, *Action*] かまたは [*Pattern*, *Condition*, *Action*]. これらの要素はすべて quote 型.

Mode 整数

- *Expr* を *Rule* に従い書き換える. *Pattern* に適合しない場合は *Expr* 自体を戻す.
- *nqt* は normalized quote の略であり *fnode* 標準形に変換してから適合検査をする. *fnode* 標準形については Section 1.2.24 [*qt_normalize*], page 18 を見よ.

```
ctrl("print_quote", 2);
nqt_match_rewrite('x*y*z', ['X*Y', 'X+Y'], 1);
  出力: (x)+((y)*(z))
A='x*x;
nqt_match_rewrite(A, ['X*Y', 'X+Y'], 1);
  出力: x^2   (マッチしていない)
nqt_match_rewrite(A, ['X*Y', 'X+Y'], 2);
  出力: 2*x
```

適合についてのモードの違いを理解するために次の例および *fnode* 標準形 (*qt_normalize*) を参照.

```
quotetolist(qt_normalize('x*x', 0));
  出力: [b_op, ^, [internal, x], [internal, 2]]
quotetolist(qt_normalize('x*x', 1));
  出力: [b_op, ^, [internal, x], [internal, 2]]
quotetolist(qt_normalize('x*x', 2));
  出力: [n_op, *, [internal, x], [internal, x]]
```

参照 Section 1.2.22 [*nqt_match*], page 17, Section 1.2.30 [*qt_rewrite*], page 22, Section 1.2.24 [*qt_normalize*], page 18

ChangeLog

- — まだ書いてない.

1.2.24 qt_normalize

`qt_normalize(Expr[, Mode])`

:: *Expr* を *fnode* 標準形に変換する. *Mode* により標準形への展開アルゴリズムを指定できる.

return quote 型

Expr quote 型

Mode 整数

`fnode` は `quote` 型の実体である。 `fnode` は木であり、型 `id` および子供からなる。型および子供を取り出す関数が `funargs_to_quote` である。また `fnode` をリストに変換する関数が `quotetolist` である。

`fnode` の標準形はパターンマッチング、書き換えを容易におこなうために導入された。 `fnode` の標準形を `fn` と書くとき、標準形の BNF 風表現での定義は以下のとおり。

```
fn          = formula | functor(nf [...]) | sum_of_monom
  fnode の標準形.  functor は関数よびだしたいなもの.
sum_of_monom = monom [+ ...]
  モノミアルの和
monom       = [formula *] nfpow [* ...]
  モノミアル
nfpow       = nf | nf^(nf)
  冪乗部分の標準形
formula     = Risa object
```

- `Expr` を `fnode` 標準形に変換する。 `Mode` により標準形への展開アルゴリズムを指定できる。
- 展開は再帰的である。
- 入力が `fnode` に変換された初期状態では `+` や `*` は子供が 2 人の binary operator (`b_op`) であるが、`qt_normalize` を作用させることにより、`+` や `*` は任意人数の子供を持てる `n`-ary operator に変換される。
- `n`-ary operator を基礎とした `fnode` 標準形を用いることにより、パターンマッチ用のパターンの数を減らせることが経験的にわかっている。
- `Mode=0`. 展開しない。これが既定の動作。
- `Mode=1`. 展開する。ただし `x*x` 等を `x^2` 等に変換
- `Mode=2`. 展開する。ただし `x*x` 等を `x^2` 等に変換しない。

`Mode` の違いについては以下の例も参考に。

```
ctrl("print_quote",2);
A=quote((x-y)*(x+y));
  出力: ((x)-(y))*((x)+(y))
B=qt_normalize(A,0);
  出力: ((x)+((-1)*(y)))*((x)+(y))  Mode=0. 展開はされない. +, * は n_op (nary-op) へ
quotetolist(B);
  出力: [n_op,*,[n_op,+, [internal,x], [n_op,*, [internal,-1], [internal,y]]], [n_op,+, [internal,x], [n_op,*, [internal,-1], [internal,y]]]]

B=qt_normalize(A,1);
  出力: ((x)^(2))+((x)*(y))+((-1)*((y)^(2)))+((-1)*(y)*(x))
  Mode=1. 展開する. +, * は n_op (nary-op) へ. 巾をまとめる.
quotetolist(B);
  出力: [n_op,+, [b_op,^, [internal,x], [internal,2]], [n_op,*, [internal,x], [internal,y]], [n_op,+, [internal,x], [b_op,^, [internal,x], [internal,2]], [n_op,*, [internal,x], [internal,y]]]]

qt_normalize(A,2);
  出力: ((x)*(x))+((x)*(y))+((-1)*(y)*(x))+((-1)*(y)*(y))
  Mode=2. 展開する. +, * は n_op (nary-op) へ. 巾は使わない.
```

```
quotetolist(B);
```

出力: $[n_op, +, [b_op, ^, [internal, x], [internal, 2]], [n_op, *, [internal, x], [internal, y]$

```
qt_normalize('x^2,2);
```

出力: $(x)*(x)$

Mode=2. 巾は使わない. n-ary の * へ.

参照 Section 1.2.22 [nqt_match], page 17, Section 1.2.23 [nqt_match_rewrite], page 18, $\langle undefined \rangle$ [quotetolist], page $\langle undefined \rangle$, Section 1.2.18 [quote_to_funargs], page 13

ChangeLog

-

1.2.25 qt_set_coef

```
qt_set_coef(ParamList)
```

:: 以下 *ParamList* に現れる多項式変数を変数とする有理関数体を係数とする非可換多項式を扱う.

return リスト

ParamList リスト

- 以下 *ParamList* に現れる多項式変数を変数とする有理関数体を係数とする非可換多項式を扱う.
- この宣言をしないと係数体を数とする非可換多項式として計算する.
- qt_normalize およびその機能を用いる関数がこの機能の影響を受ける.
- qt_comp 関数がこの機能の影響を受ける.

```
ctrl("print_quote",2);
```

```
qt_set_coef([a]);
```

```
B=qt_normalize(quote((a*x+a)^2),2);
```

出力: $((a^2)*(x)*(x))+((2*a^2)*(x))+(a^2)$

```
qt_normalize(B+B,2);
```

出力: $((2*a^2)*(x)*(x))+((4*a^2)*(x))+(2*a^2)$

参照 Section 1.2.24 [qt_normalize], page 18

ChangeLog

- —まだ書いてない.

1.2.26 qt_set_ord

```
qt_set_ord(VarList)
```

:: *VarList* を変数順序とする.

return リスト

VarList リスト

- *VarList* を辞書式に用いた変数順序を以下使用する.
- この宣言をしないとある不定元についての既定の辞書式順序——まだ書いてない——を用いて項を比較する. *VarList* に現れない変数についてはこの順序が適用される.

- `qt_normalize` およびその機能を用いる関数がこの機能の影響を受ける.
- `qt_comp` 関数がこの機能の影響を受ける.

```
ctrl("print_quote",2);
qt_normalize(quote(x+y),2);
    出力: (x)+(y)
qt_set_ord([y,x]);
    出力: [y,x,z,u,v,w,p,q,r,s,t,a,b,c,d,e,f,g, 以下省略 ]
qt_normalize(quote(x+y),2);
    出力: (y)+(x)
```

参照 Section 1.2.24 [`qt_normalize`], page 18, Section 1.2.28 [`nqt_comp`], page 22

ChangeLog

- — まだ書いてない.

1.2.27 `qt_set_weight`

`qt_set_weight(WeightVector)`
 :: 変数について `weight` ベクトルを設定する.

return リスト

WeightVector
 リスト

`fnode f` の `weight w(f)` は次の式で計算する.

`f` が葉の場合は原則 0. `qt_weight_vector` で `weight` が与えられている不定元についてはその値.

`f` が `node` の場合は次の規則で再帰的にきめる.

$$w(f+g) = \max(w(f), w(g))$$

$$w(f \ g) = w(f) + w(g)$$

$$w(f^n) = n \ w(f)$$

関数については? -----まだ書いてない.

- *WeightVector* でまず順序の比較をして, それから `qt_set_order` による順序, 最後に既定の順序で比較する. *WeightVector* に現れない変数についての `weight` は 0 となる.
- `qt_normalize` およびその機能を用いる関数がこの機能の影響を受ける. `qt_normalize` で の展開では, この順序を用いて項がソートされる.
- `qt_comp` およびその機能を用いる関数がこの機能の影響を受ける.
- `weight` ベクトルによる順序比較についてはグレブナ基底の節 Section 1.2.5 [`dp-gr-main`], page 4 も参照.

```
ctrl("print_quote",2);
qt_set_weight([[x,-1],[y,-1]]);
    結果: [[x,-1],[y,-1]]
qt_normalize(quote( 1+(x+y)+(x+y)^2),1);
    結果: (1)+(y)+(x)+((y)^(2))+((y)*(x))+((x)^(2))+((x)*(y))
```

参照 Section 1.2.24 [`qt_normalize`], page 18, Section 1.2.26 [`qt_set_ord`], page 20,
 Section 1.2.27 [`qt_set_weight`], page 21, Section 1.2.5 [`dp-gr-main`], page 4

ChangeLog

-

1.2.28 nqt_comp

`nqt_cmp(Expr1, Expr2)`

:: *Expr1* と *Expr2* の順序を比較する.

return 整数

Expr1, Expr2

quote 型

- *Expr1* と *Expr2* の順序を比較する.
- *Expr1* > *Expr2* なら 1.
- *Expr1* < *Expr2* なら -1.
- *Expr1* = *Expr2* (おなじ順序) なら 0.

```
ctrl("print_quote",2);
```

```
qt_set_ord([y,x]); qt_set_weight([[x,-1],[y,-1]]);
```

```
[nqt_comp('x','y'), nqt_comp('y','x'), nqt_comp('x','x')];
```

出力: [-1,1,0]

参照 Section 1.2.24 [qt_normalize], page 18, Section 1.2.26 [qt_set_ord], page 20,
Section 1.2.27 [qt_set_weight], page 21

ChangeLog

- まだ書いてない.

1.2.29 qt_is_var, qt_is_coef

`qt_is_var(Expr)`

:: *Expr* が不定元に対応する quote なら 1 を戻す.

`qt_is_coef(Expr)`

:: *Expr* が係数の有理関数体に属するとき 1 を戻す.

return 整数

Expr quote 型

- *Expr* が不定元に対応する quote なら 1 を戻す. そうでないとき 0 を戻す.

```
[qt_is_var(quote(x)), qt_is_var(quote(3/2))];
```

出力: [1,0]

参照 Section 1.2.30 [qt_rewrite], page 22, Section 1.2.23 [nqt_match_rewrite], page 18

ChangeLog

- — まだ書いてない.

1.2.30 qt_rewrite

`qt_rewrite(Expr, Rules, Mode)`

:: *Expr* を規則集合 *Rules* を用いて書き換える.

return quote 型

Expr quote 型

Rules リスト

Mode 整数

- ユーザ言語を用いて定義された関数. `import("norow_rewrite.rr")` しておくこと. (`norow_rewrite.rr` が `OpenXM/lib/asir-contrib` に存在しない場合ソースの `OpenXM/src/asir-contrib/testing/norow/new_rewrite.rr` をコピー)
- *Expr* を規則集合 *Rules* を用いて書き換える.
- 規則の適用は `fnode` 木に対して再帰的である. 一方 `nqt_match_rewrite` ではトップレベルのみに規則が適用される.
- 規則集合 *Rules* の各要素の書き方は `nqt_match()` の *Pattern* と同じ書き方. つまり [パターン, 書き換え結果] または [パターン, 条件, 書き換え結果].
- *Mode* の意味は Section 1.2.24 [`qt_normalize`], page 18 の *Mode* と同様. パターンマッチ, 書き換えは *Mode* で `qt_normalize()` されてから遂行される.

注意: 数学的には $X*Y=Y*X$ が可換性を与える規則だが, これをそのまま規則として与えると書き換えが停止しない. 次の例では, 上の例のように順序比較し, たとえば, 順序が大きくなる場合のみに書き換えるべきである.

```
import("norow_rewrite.rr");
R=[['X*Y','nqt_comp(Y*X,X*Y)>0','Y*X]];
qt_rewrite('(x-y)^2,R,2);
出力: quote(x*x+-2*x*y+y*y)
```

外積代数の計算 (`asir-contrib` をロードした状態).

```
import("norow_rewrite.rr");
Rext0=[quote(X*Y),quote(qt_is_var(X) && qt_is_var(Y) && nqt_comp(Y,X)>0),
quote(-Y*X)];
Rext1=[quote(X^N),quote(eval_quote(N)>=2),quote(0)];
Rext2=[quote(X*X),quote(0)];
Rext=[Rext0,Rext1,Rext2];
qt_rewrite(quote((x+2*y)*(x+4*y)), Rext,1);
出力: 2*x*y

qt_set_coef([a,b,c,d]);
qt_rewrite(quote((a*x+b*y)*(c*x+d*y)), Rext,1);
出力: (d*a-c*b)*x*y
```

微分の計算 (`asir-contrib` をロードした状態).

```
import("norow_rewrite.rr");
qt_set_coef([a,b]);
Rd1=['d(X+Y)','d(X)+d(Y)'];
Rd2=['d(X*Y)','d(X)*Y+X*d(Y)'];
Rd3=['d(N)','qt_is_coef(N)','0'];
Rd4=['d(x)','1'];
Rd=[Rd1,Rd2,Rd3,Rd4];
B=qt_rewrite('d((a*x+b)^3),Rd,2);
```

出力: `quote(3*a^3*x*x+6*b*a^2*x+3*b^2*a)`
`fctr(eval_quote(B));`
 出力: `[[3,1],[a,1],[a*x+b,2]]`

参照 Section 1.2.22 [nqt_match], page 17, Section 1.2.23 [nqt_match_rewrite], page 18, Section 1.2.24 [qt_normalize], page 18

ChangeLog

- qt 系の関数の原型は OpenXM/src/asir-contrib/testing/tr.rr である. このユーザ言語による開発が 2005 年の春まで行われ, そのあと組み込み関数主体の qt 系の関数が開発された.
- qt 系の関数についてのその他の参考文献: OpenXM/doc/Papers/2005-rims-noro.tex および OpenXM/doc/Papers/2005-rims-noro.tm (TeXmacs の記事).
- Todo: qt 系の関数を用いたおもしろい計算を Risa/Asir ジャーナルの記事として書く.

1.2.31 xyz_pqr, syz_stu

`xyz_pqr(arg1, arg2[, flag])`
`xyz_stu(arg1, arg2)`
 :: xyz に関する操作.

return 整数

arg1, arg2 整数

flag 0 または 1

- この項目は新しい関数の説明を書くためのテンプレートである. 消すな.
- `xyz_pqr()` は, *arg1*, *arg2* を pqr する.
- *flag* が 0 でないとき, モジュラ計算を行う.
- `xyz_stu()` は stu アルゴリズムを用いる.

```
[219] xyz_pqr(1,2);
3
[220] xyz_pqr(1,2,1);
3
0
[221] xyz_stu(1,2);
3
```

参照 `<undefined>` [xyz_abc], page `<undefined>`

ChangeLog

- この関数は 2004-3-1 から 2004-3-14 にかけて アルゴリズム xyz (論文 <http://www.afo.org/xyz.pdf>) を用いて書き直された. 変更をうけたソースコードは xxxyy.rr, ppp.c である.
- この関数は 2000 頃にはじめてのバージョンが書かれた. ソースは ppp.c である.

Index

(Index is nonexistent)

(Index is nonexistent)

Short Contents

1	実験的仕様の関数	1
	Index	25

Table of Contents

1	実験的仕様の関数	1
1.1	実験的仕様の関数説明書について	1
1.2	実験的仕様の関数	1
1.2.1	quotetotex, quotetotex_env	1
1.2.2	objtoquote	2
1.2.3	copyright	3
1.2.4	string_to_tb, tb_to_string, write_to_tb	3
1.2.5	dp_gr_main	4
1.2.6	asir-port.sh, asir-install.sh	5
1.2.7	get_struct_name, get_element_names, get_element_at, put_element_at	6
1.2.8	dp_weyl_gr_main	7
1.2.9	dp_initial_term	8
1.2.10	dp_order	9
1.2.11	mapat	9
1.2.12	list	10
1.2.13	set_print_function	10
1.2.14	small_jacobi	11
1.2.15	flatten_quote	11
1.2.16	assoc	12
1.2.17	sprintf	12
1.2.18	quote_to_funargs, funargs_to_quote, remove_paren..	13
1.2.19	set_secure_flag, set_secure_mode	14
1.2.20	initialize_static_variable	15
1.2.21	eval_quote	16
1.2.22	nqt_match	17
1.2.23	nqt_match_rewrite	18
1.2.24	qt_normalize	18
1.2.25	qt_set_coef	20
1.2.26	qt_set_ord	20
1.2.27	qt_set_weight	21
1.2.28	nqt_comp	22
1.2.29	qt_is_var, qt_is_coef	22
1.2.30	qt_rewrite	22
1.2.31	xyz_pqr, syz_stu	24
	Index	25