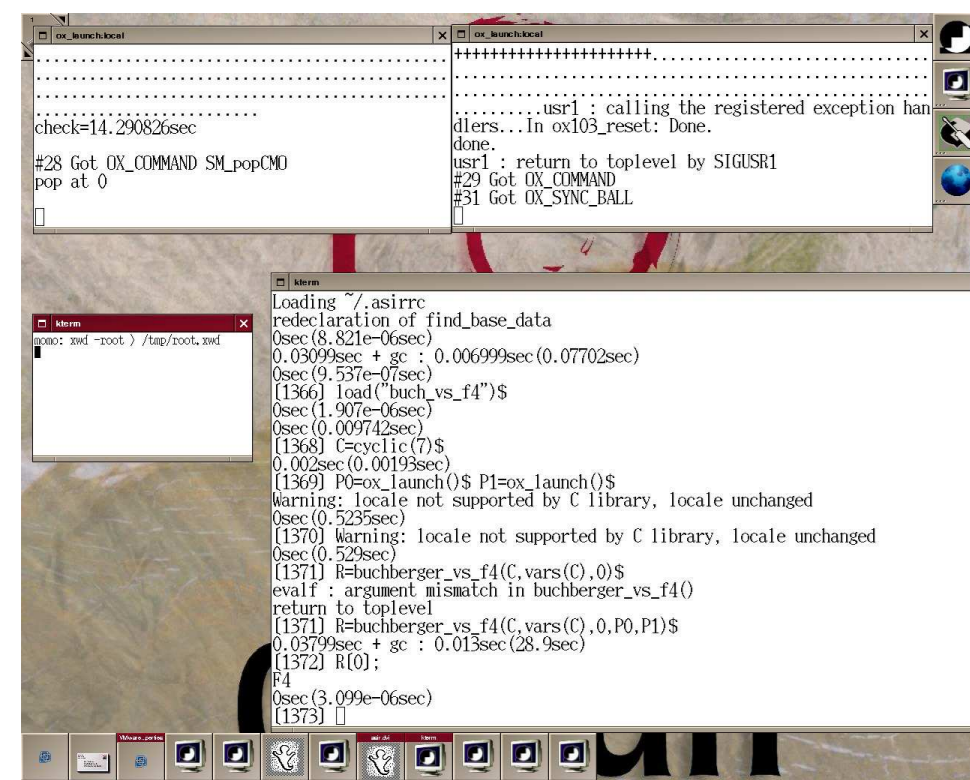


SOME USEFUL FUNCTIONS IN RISA/ASIR

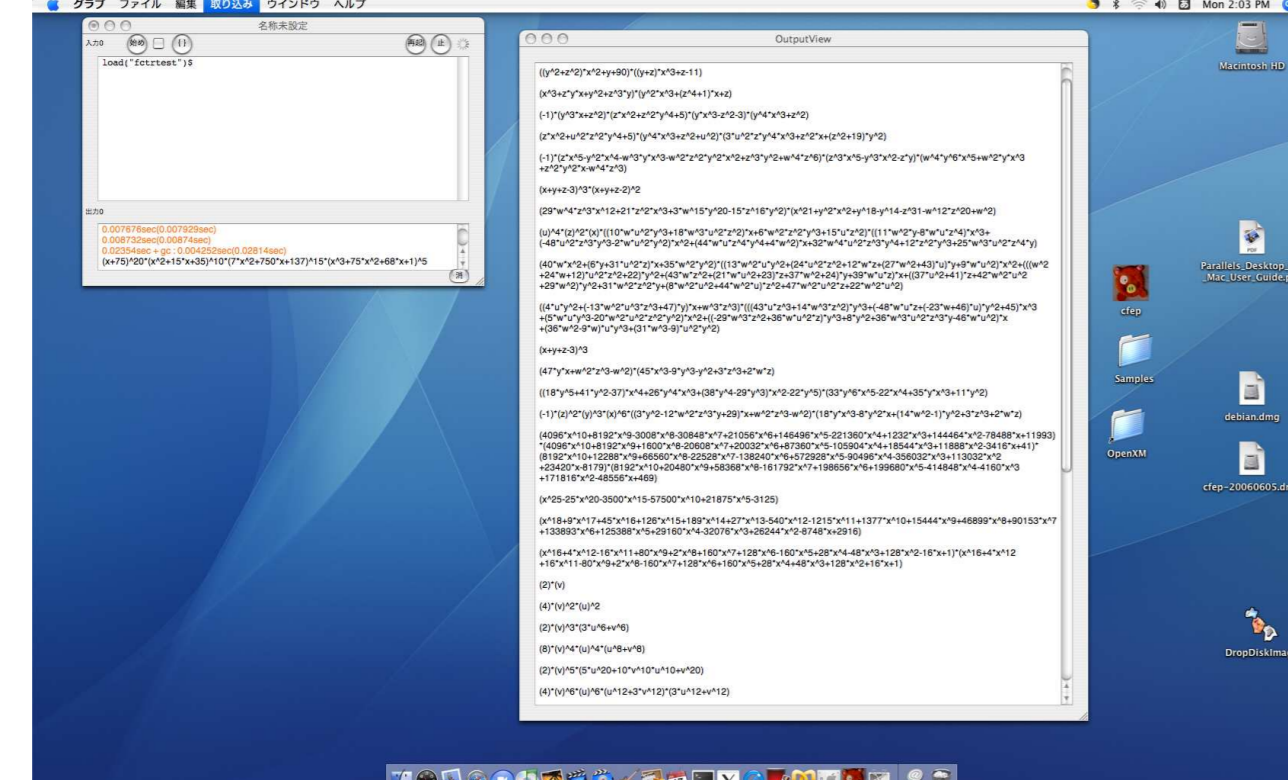
Masayuki Noro and Nobuki Takayama (Kobe University, Japan)

What is Risa/Asir?

- Old style software for polynomial computation
- User language with C-like syntax
- Open source
- Intensive use of modular computation
Groebner basis computation, modular change of ordering, modular dynamic evaluation, ...
- Easy interface for distributed computation under OpenXM



Distributed computation



cfep : a front end for Macintosh

Availability

Risa/Asir is contained in the OpenXM package.

- Source
<http://www.math.kobe-u.ac.jp/OpenXM/Current/index.html>.
- Linux binary
<http://www.math.kobe-u.ac.jp/pub/OpenXM/head/knoppix/>
- Macintosh binary
<http://www.math.kobe-u.ac.jp/cfep/> (a front end for OpenXM components)
- Windows binary
<http://www.math.kobe-u.ac.jp/Asir/asir.html> (standalone)

New functions for Groebner Basis Computation

Efficient implementation based on homogenization and modular trace algorithm

- Trace algorithm
Skip an S-poly reduction over \mathbb{Q} if it is reduced to zero over a finite field.
⇒ The result must be checked, but it is relatively easy.
- Frequent inter-reduction in homogeneous cases
It improves efficiency of normal form computation over \mathbb{Q} .
- Options
 $Trace = 1$: trace algorithm with checking; $Trace = -1$: trace algorithm without checking
 $Homo = 1$: homogenizing algorithm with frequent inter-reduction
 $Char$ for non-trace algorithm : the characteristic of the coefficient field

Buchberger Algorithm and F_4 Algorithm

- `nd_gr`(*PolyList*, *VarList*, *Char*, *Order*)
Non-trace implementation of Buchberger algorithm
- `nd_gr_trace`(*PolyList*, *VarList*, *Homo*, *Trace*, *Order*)
Buchberger trace algorithm
- `nd_f4`(*PolyList*, *VarList*, *Char*, *Order*)
Non-trace implementation of F_4 algorithm
- `nd_f4_trace`(*PolyList*, *VarList*, *Homo*, *Trace*, *Order*)
 F_4 trace algorithm (new) : a hybrid algorithm of Buchberger and F_4 with trace

```
[0] load("cyclic")$ C=cyclic(7)$ V=vars(C)$ cputime(1)$
[10] [11] [12] 0sec(7.153e-06sec)
[13] nd_gr_trace(C,V,1,1,0)$ /* GB computation of cyclic-7 */
```

```
33.82sec + gc : 5.574sec(39.4sec)
[14] nd_f4_trace(C,V,1,1,0)$
22.81sec + gc : 4.539sec(27.37sec) /* the fastest one */
[15] H=map(homogenize,C,V,h)$ /* homogenization */
0sec(0.000252sec)
[16] nd_f4(H,append(V,[h]),0,0)$
46.43sec + gc : 8.078sec(54.52sec)
[19] nd_gr(H,append(V,[h]),0,0)$
74.51sec + gc : 27.4sec(101.9sec)
```

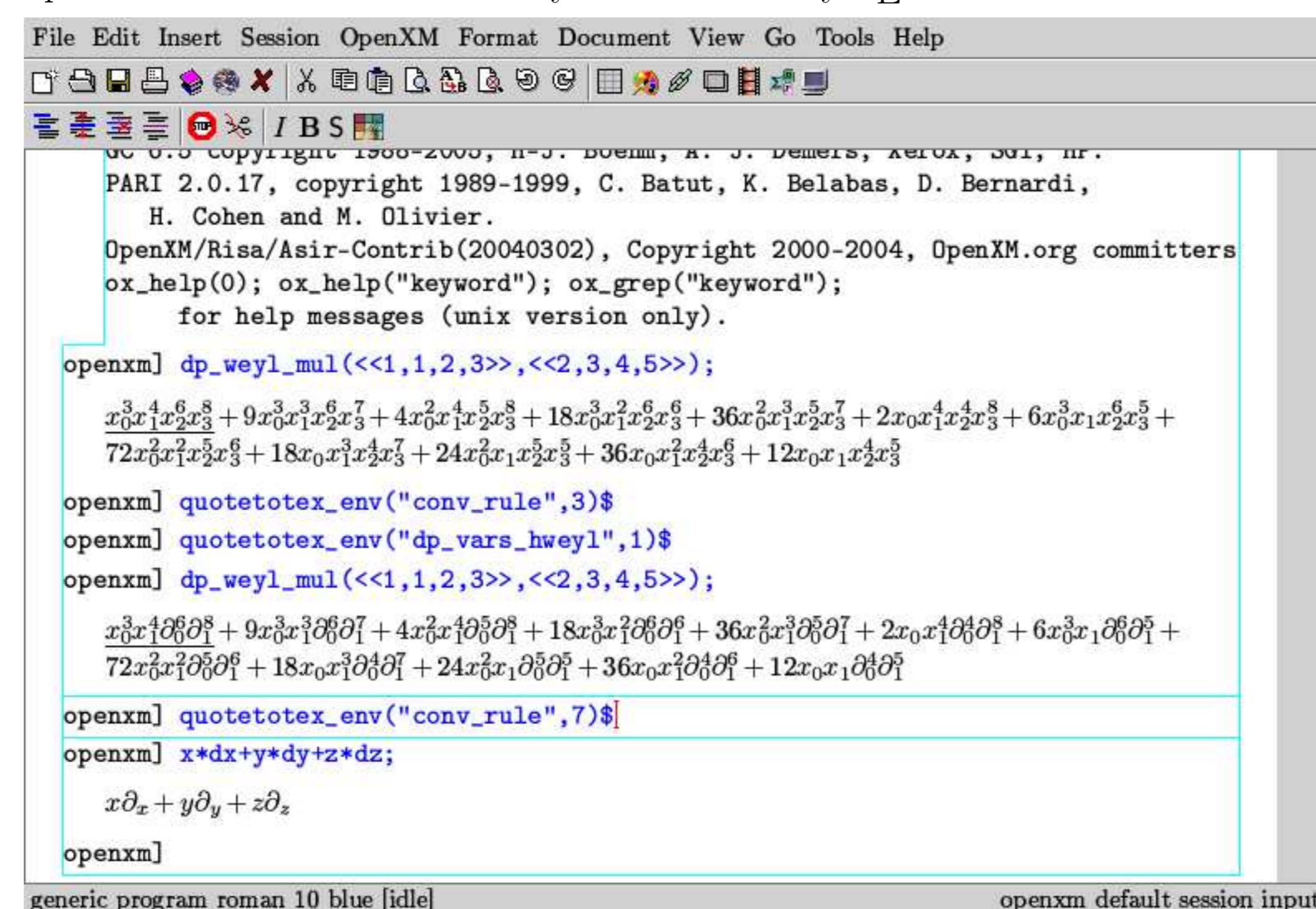
Primary ideal decomposition

- `primdec`(*PolyList*, *VarList*) `lib/primdec`
Primary ideal decomposition over \mathbb{Q} . Output is a list $[[Q_1, P_1], [Q_2, P_2], \dots, [Q_m, P_m]]$, where Q_i is a primary ideal and P_i is its associated prime.
- `primdec`(*PolyList*, *VarList*) `in lib/primdec`
Computation of minimal associated primes over \mathbb{Q}
- `primdec_mod`(*PolyList*, *VarList*, *Order*, *P*, *Strategy*) `lib/primdec_mod`
Computation of minimal associated primes over a small prime field \mathbb{F}_P ($P < 2^{29}$)
Order : a hint (usually set to 0); *Strategy* = 1 : early termination (not necessarily fast)

```
[0] load("primdec");
[181] F = [x+(-2*z+u)*y-1, (2*z+1)*x-4*u*z+6, -y*x+2*u*z-3, x^2,
(-u-3)*x+4*u^2*z-6*u, (3*u-1)*x+(4*u^2-12)*y-4*u*z-4*u+6,
(-u^2-u)*x+12*u*z-18, (-u-1)*x+8*u*z^2-12*z]$
[182] V = [x,y,z,u]$
[183] primdec(F,V);
[[2*u*z-3, (u^2-3)*y-u, x, (2*z-u)*y+1]] /* one isolated prime */
[184] primdec(F,V);
[[[x, 2*u*z-3, (2*z-u)*y+1, (u^2-3)*y-u], [2*u*z-3, (u^2-3)*y-u, x, (2*z-u)*y+1]],
[[x-12*y+(-16*u+12)*z+30, 11*x+(16*u+12)*y+36*z-8*u^2-16*u-6, ...],
[u^3+u^2-3*u-9, 6*z-u^2-u+3, 6*y-u^2-u, x]]] /* one embedded component */
```

Computation in Weyl Algebra

$x_1^{i_1} \dots x_n^{i_n} \partial_1^{j_1} \dots \partial_n^{j_n} \in D = \mathbb{Q}\langle x_1, \dots, x_n, \partial_1, \dots, \partial_n \rangle$: represented by $\langle (i_1, \dots, i_n, j_1, \dots, j_n) \rangle$
 $P_1 \cdot P_2$ in D is computed by `dp_weyl_mul(P1, P2)`, but hard to read.
⇒ The output can be converted to an easy-to-read form by `TEX` macros.



Computation of Groebner bases and b-functions

- `nd_weyl_gr`(*PolyList*, *VarList*, *Char*, *Order*)
Buchberger algorithm in Weyl algebra
(the old one: `dp_weyl_gr_main(PolyList, VarList, Homo, Trace, Order)`)

```
[0] nd_weyl_gr([x*dx^2+y*dx*dy, y*dx*dy-x*dy^2+x], [x,y,dx,dy], 0, 0);
[-dy^2+1, dx]
```

- `bfunction`(*F*) `in lib/bfct`
An improved algorithm for computing the *b*-function of *F*
- `generic_bfct`(*PolyList*, *VarList*, *DVarList*, *Weight*) `in lib/bfct`
The *b*-function of a holonomic *D*-ideal $I = \langle PolyList \rangle$ with respect to a weight $w = Weight$ (the monic generator of $\text{in}_{(-w,w)}(I) \cap \mathbb{Q}[s]$, $s = w_1x_1\partial_1 + \dots + w_nx_n\partial_n$)

```
[0] load("bfct")$ computation of b-function
[216] F=x^4+x*y^4+y^5;
[217] B=bfunction(F);
-2560000000000000*s^13-307200000000000*s^12-168960000000000*s^11-...
[218] B1=generic_bfct([F-t,dx+diff(F,x)*dt,dy+diff(F,y)*dt],[t,x,y],
[dt,dx,dy],[1,0,0]);
256000000000000*s^13+256000000000000*s^12-63360000000000*s^10-...
[219] B-subst(B1,s,-s-1); /* these two must be equal */
0
```

Computation over Algebraic Number Fields

- Roots
A root of *F* is generated by `newalg(F)`, where *F* is a univariate polynomial with coefficients containing predefined roots. ⇒ multiple (successive) algebraic extension
- Multiple algebraic extension $K = \mathbb{Q}(\alpha_k, \dots, \alpha_1)$
Represented by a list *AlgList* = $[\alpha_k, \dots, \alpha_1]$, where α_i is defined over $\mathbb{Q}(\alpha_{i-1}, \dots, \alpha_1)$.

Univariate factorization and splitting Fields

- `af`(*F*, *AlgList*) `in lib/sp`
Factorize a univariate polynomial *F* over $\mathbb{Q}(\alpha_k, \dots, \alpha_1)$.
- `sp`(*F*) `in lib/sp`
Compute the splitting field of *F* and the set of linear factors of *F* over the field.
[FactorList, RootList] is returned.

```
[0] load("sp")$ factorization and splitting field
[101] af(x^4+4*x^3+5*x^2+2*x+1,[newalg(t^2+1)]); /* over Q(sqrt(-1)) */
[[1,1],[x^2+(#0+2)*x+(#0),1],[x^2+(-#0+2)*x+(-#0),1]]
[102] sp(x^5-3*x^2+2*x+1);
[[x+(-#2), 5*x+(5*#2+#1^4-...), 5*x+((#1^4+2*#1^2-#1-2)*#2-...),
5*x+((-#1^4-2*#1^2+#1+2)*#2+...), x+(-#1)], /* linear factors */
[[(#2), 5*t#2^2+(t#1^4-3*t#1^3-...), /* root #2 and its defining poly */
(#1), t#1^5-3*t#1^2+2*t#1+1]] /* root #1 and its defining poly */
```

Groebner Basis Computation

- `nd_gr_trace`(*PolyList*, *VarList*, *Homo*, *Trace*, *Order*)
If *PolyList* contains roots $\alpha_k, \dots, \alpha_1$, then the Groebner basis of $\langle PolyList \rangle$ is computed over $\mathbb{Q}(\alpha_k, \dots, \alpha_1)$

```
[0] load('cyclic')$ C=cyclic(7)$ V=[c0,c1,c2,c3,c4,c5]$
[10] [11] [12] A=newalg(t^6+t^5+t^4+t^3+t^2+t+1)$
[13] G=nd_gr_trace(map(subst,C,c6,A),V,1,1,0)$
[14] G[0];
c0+c1+c2+c3+c4+c5+(#0)
[15] G[1];
c1^2+(c3+c4+c5+(2*#0))*c1+(-c3+(#0))*c2+(-c4+(#0))*c3+(-c5+(#0))*c4+(#0^2)
[16] G[2];
((152402*c5+(702570*#0))*c2+(690392*c4-328074*c5+(103596*#0))*c3-...
```