Concetti essenziali ed esempî d'uso di CoCoA

Daniele A. Gewurz* Francesca Merola

CoCoA è un programma che permette di fare calcoli di algebra commutativa. Il nome viene da "*Co*mputations in *Co*mmutative *A*lgebra. La versione più recente (che al momento in cui scriviamo è la 4.7, con le sue sottoversioni) è disponibile liberamente in rete all'indirizzo

`http://cocoa.dima.unige.it`.

Avvio. Il programma viene avviato digitando `cocoa` dalla riga di comando o, per le versioni con interfaccia grafica (WinCoCoA, MacCoCoA), facendolo partire come qualunque altro programma dall'opportuna icona o menu.

Come si usa. I comandi di CoCoA terminano con il punto e virgola; inoltre CoCoA è "case-sensitive", cioè considera distinte lettere maiuscole e minuscole. In particolare, i comandi e le variabili cominciano sempre con la lettera maiuscola, e le minuscole sono riservate alle indeterminate.

Nella versione con interfaccia grafica, per eseguire un qualsiasi comando lo si digita nello spazio "interattivo" - che in genere è la metà finestra in basso - e lo si manda in esecuzione con control-invio (sui Mac con command-invio). Il comando e il suo effetto appariranno nella parte superiore della finestra, mentre ciò che è stato digitato scomparirà in basso. È possibile aprire in basso altre sottofinestre ("tab") non interattive in cui lavorare, per esempio per scrivere un programma da testare e debuggare via via.

Per ottenere il "tutorial" (introduzione guidata tramite esempi) si può dare il comando `?tutorial` dall'interno di CoCoA oppure, in alcune installazioni, scegliere dal menu di "Help". Inoltre è possibile chiedere informazioni su una specifica istruzione o argomento con un comando della forma `?GCD` (con cui si ottiene una spiegazione dell'istruzione `GCD`) o `?divisor` (con cui si ottiene un elenco delle istruzioni che hanno a che fare con divisori). Con il solo `?` si accede alla schermata iniziale sulla documentazione.

Salvare e leggere file. CoCoA permette di salvare in vari modi quello che si sta facendo. Se si sta lavorando a un programma, in genere conviene farlo in una finestra "non interattiva" (una di quelle che si aprono dopo quella già pronta all'apertura del programma) e usare il comando "Save" dal menu. Se si vuole salvare tutto quello che è comparso finora nella finestra superiore si può usare il comando "Save Output As...".

In ogni caso i file ottenuti saranno di solo testo, e quindi si potranno leggere e (con cautela) modificare con qualsiasi editor di testi.

(Per favore **non** scrivete i programmi in formato Word e simili.)

Operazioni e strutture fondamentali. CoCoA riconosce le operazioni standard `+`, `-`, `*` (moltiplicazione; quando non c'è ambiguità si può omettere: per esempio `xy` è equivalente a `x*y`), `/` (divisione), `^` (elevamento a potenza) tra numeri (interi, razionali o interi modulo un intero N) e tra polinomi.

```
P := 3xy+7x^2z;
Q := x^5z + 2/3y^2;
P + Q;
x^5z + 7x^2z + 3xy + 2/3y^2
------------------------------
P*Q;
```

*gewurz@mat.uniroma1.it

```
7x^7z^2 + 3x^6yz + 14/3x^2y^2z + 2xy^3
------------------------------
P^2;
49x^4z^2 + 42x^3yz + 9x^2y^2
------------------------------
```

Inoltre permette di calcolare il massimo comun divisore (`GCD`, per Greatest Common Divisor), il minimo comune multiplo (`LCM`, per Least Common Multiple), il quoziente col resto (per i numeri interi i comandi sono `Div` per il quoziente e `Mod` per il resto, per i polinomi vedi oltre).

CoCoA "conosce" gli anelli `Z` (interi) e `Q` (razionali), ma non il campo dei numeri reali (e neanche complessi); inoltre gli anelli `Z/(N)` (interi modulo N) e gli anelli di polinomi, che si definiscono con istruzioni della forma `R::=Q[x,y,z]`. Alla partenza, il programma lavora con l'anello `Q[x,y,z]`; per cambiare l'"anello base" si usa l'istruzione `Use`. Per esempio, il comando

```
Use R::=Q[a,b,c];
```

fa sì che d'ora in avanti si lavorerà nell'anello $\mathbb{Q}[a, b, c]$.

Per visualizzare l'anello con cui si sta attualmente lavorando si può dare l'istruzione `CurrentRing();`[1] per avere la sola lista delle indeterminate si usa `Indets()`.

Per usare un oggetto definito in un certo anello dopo che si è passati a un altro anello si usa l'istruzione `BringIn`. Per verificare in quale anello era stato definito un oggetto, poniamo F, si può dare il comando `RingEnv(F)`.

Liste. I vari oggetti con cui CoCoA lavora possono essere articolati in liste, rendendo alcuni calcoli più semplici, potendo simultaneamente compiere la stessa operazione su tutti gli elementi della lista, o accedere a quello che serve in quel momento. Definiamo una lista di numeri:

```
L := [2,4,6,8];
```

Aggiungiamo un elemento:

```
Append(L, 10);
```

Ora la lista L sarà uguale a $[2, 4, 6, 8, 10]$. Modifichiamo un elemento:

```
L[2] := 7;
```

Adesso L sarà diventata $[2, 7, 6, 8, 10]$. Se vogliamo calcolare i quadrati di tutti gli elementi della lista, possiamo usare una notazione molto simile a quella comunemente usata per definire un insieme:

```
[X^2 | X In L];
```

ottenendo $[4, 49, 36, 64, 100]$.

Per rimuovere un elemento si può usare l'istruzione `Remove`: per esempio, con `Remove(L, 3)` si elimina il terzo elemento della lista L, ottenendo $[2, 7, 8, 10]$. Al contrario, per aggiungere un elemento, oltre che `Append`, si può usare `Insert` per inserirlo in una posizione desiderata. Quindi `Insert(L, 4, 22)` inserirà nella quarta posizione l'elemento 22, facendo diventare la nostra lista $[2, 7, 8, 22, 10]$.

Sul manuale, o usando le funzioni di aiuto interne a CoCoA, si possono vedere le numerose altre funzioni e modi per utilizzare le liste (per esempio, tra le tante, l'operatore `Concat` per concatenare due liste, l'operatore `><` per definire il prodotto cartesiano e l'operatore `..` per definire un intervallo, quindi `1..5` è equivalente a `[1, 2, 3, 4, 5]`).

Esercizio 1 *Definire una lista di almeno cinque elementi, i cui termini siano a loro volta liste, di lunghezza almeno tre. Come si fa ad accedere al terzo elemento della quinta lista? E a costruire una nuova lista di lunghezza tre il cui l'i-mo termine sia la somma dei termini dell'i-ma lista di partenza? (Può essere utile il comando* `Sum`*.)*

Esercizio 2 *Cercare come si usano le funzioni* `Head` *e* `Tail`*.*

[1] Si noti che qui e altrove si usano funzioni con zero argomenti.

Programmare CoCoA. CoCoA incorpora un proprio linguaggio di programmazione che permette di automatizzare i calcoli e di definire algoritmi.

Per la descrizione completa del linguaggio si veda il manuale; qui ci limiteremo ad alcuni esempi. Chi ha un po' di familiarità con il Pascal o il C dovrebbe riconoscere le strutture generali, come i cicli `For` (e `Foreach` per definire un ciclo su tutti gli elementi di una lista data) e `While`, le espressioni condizionali con `If` ... `Then` e così via. Si tenga solo a mente che in CoCoA ognuna di queste costruzione è conclusa da un comando "End" corrispondente (`EndFor`, `EndWhile` etc.). Per esempio, la funzione

```
Define F(A,B)
   If A > B Then
      Return A-B;
   EndIf;
   Return B-A;
EndDefine;
```

calcola il modulo della differenza fra `A` e `B`. Si noti che in CoCoA non è necessario dichiarare le variabili all'interno di un programma o della definizione di una funzione.

Esercizio 3 *(Facoltativo) In genere nelle funzioni di CoCoA gli argomenti sono passati* per valore*, ma è possibile anche passarli* per riferimento *con il comando* `Var`*. Studiare come funziona e scrivere due funzioni che riordinano una lista di numeri: in una la lista viene passata per valore e la funzione dà in output un'altra lista composta con gli elementi della prima ma riordinati; nella seconda funzione la lista viene passata per riferimento e la funzione trasforma la lista data.*

Un altro esempio di implementazione di un famoso algoritmo è

```
Define Euclide(A, B)
    While B <> 0 Do
        X := Mod(A, B);
        A := B;
        B := X;
    EndWhile;
    Return A;
EndDefine;
```

che permette di ottenere il massimo comun divisore fra due interi con l'algoritmo euclideo.

Esercizio 4 *Scrivere lo stesso algoritmo per i polinomi in una variabile.*

Poniamo di voler implementare l'algoritmo per la divisione tra un polinomio e (i generatori di) un ideale. Questa operazione è già definita in CoCoA con l'operatore `DivAlg`, come abbiamo visto, ma qui la useremo come esempio.

```
Define LMDivide(H,K)
   D := DivAlg(LM(K), [LM(H)]);
   If D.Remainder = 0 Then
      Return True
   Else
      Return False
   EndIf
EndDefine;

Define Divisioni(F,G)
     S := Len(G); A := NewList(S,0);
     R :=0; P:=F;
     While P<>0 Do
          I := 1; Occur := False;
```

```
          While I<=S And Occur=False Do
               If LMDivide(LM(G[I]), LM(P)) Then
                    A[I] := A[I] + LM(P)/LM(G[I]);
                    P := P - (LM(P)/LM(G[I]))*G[I];
                    Occur := True
               Else
                    I := I+1
               EndIf;
          EndWhile;
          If Occur=False Then
               R := R + LM(P);
               P := P - LM(P)
          EndIf;
     EndWhile;
Return [A, R]
EndDefine;
```

Questo ovviamente non è un programma vero e proprio, ma solo la definizione di due funzioni. Di queste, la seconda, che fa uso nella sua definizione della prima, riceve in *input* un polinomio (`F`) e una lista di polinomi (`G`) e dà in *output* i quozienti (`A`) e il resto (`R`).

Esercizio 5 *Nella definizione della funzione* `LMDivide` *si è un po' barato, perché si è usato l'operatore* `DivAlg`*. Scrivere almeno un altro paio di modi più "onesti" per definire una funzione che individui se un monomio ne divide un altro.*

(Suggerimenti: studiare i "tipi" di oggetti noti a CoCoA - si vedano le funzioni `Type` *e* `Types` *e capire che tipo deve avere K/H affinché H divida K; oppure trovare gli esponenti delle varie indeterminate in H e K con le funzioni* `Deg` *o* `Log` *e confrontarli.)*

Funzioni per lavorare con i polinomi. CoCoA fornisce molte funzioni per lavorare con i polinomi. Oltre alle operazioni fondamentali già accennate, il comando `DivAlg` serve a eseguire la divisione col resto fra polinomi. Il suo primo argomento è un polinomio, mentre il secondo è una **lista** di polinomi (eventualmente composta da un solo polinomio).

Volendo effettuare la divisione col resto di un polinomio `F` per un polinomio `G`, si può procedere come segue:

```
F := xˆ2y+xyˆ2+yˆ2;
G := xy-1;
L := [G];
DivAlg(F,L);
Record[Quotients = [x + y], Remainder = yˆ2 + x + y]
```

L'output del comando `DivAlg` è un "record" (insieme di dati articolato in più campi) i cui campi sono: una lista di quozienti (uno solo nell'esempio precedente, perché la lista dei divisori è formata da un solo polinomio) e un resto.

Se invece dividiamo un polinomio `F` per una lista formata da (ad esempio) 3 polinomi `G1, G2, G3`, il comando `DivAlg` esegue le seguenti divisioni

```
F  = Q1*G1 + R1
R1 = Q2*G2 + R2
R2 = Q3*G3 + R3
```

e, nel record, riporta la lista ordinata dei quozienti `Q1, Q2, Q3` e l'ultimo resto `R3`. Vediamo un esempio:

```
F := xˆ5yˆ2 + 6xˆ3yˆ3 + yˆ4;
G1 := xy-1;
G2 := yˆ2-1;
```

```
G3 := x^3y-2;
L := [G1, G2, G3];
DivAlg(F,L);
Record[Quotients = [x^4y + 6x^2y^2 + x^3 + 6xy + 6, y^2 + 1, 0],
Remainder = x^3 + 7]
```

Altre operazioni permettono, dato un polinomio, di estrarne vari dati. Per esempio:

- `Support` dà la lista dei monomi che compongono il polinomio - tutti con coefficiente 1 - ordinati in base all'ordinamento monomiale in corso (vedi oltre per gli ordinamenti): se si sta usando `DegRevLex`, l'istruzione `Support(x^2y+3yz-2y^3)` dà `[x^2y, y^3, yz]`. `Monomials` dà invece la lista ordinata dei monomi muniti del loro coefficiente.
- `Coefficients` dà la lista dei coefficienti ordinati analogamente: `Coefficients(x^2y+3yz-2y^3)` fornisce la lista `[1, -2, 3]`.
- `Deg`, applicato a un monomio, ne dà il grado complessivo: `Deg(x^3yz^2)` dà come risultato 6. Gli si può dare opzionalmente come secondo argomento una variabile; in questo caso dà il grado rispetto a quella variabile: `Deg(x^3yz^2, x)` dà 3. Applicato a un polinomio, `Deg` dà il grado del monomio di grado massimo o, con l'argomento opzionale, il massimo grado a cui compare la variabile.

Esercizio 6 *Studiare, e implementare come funzione in CoCoA, l'algoritmo per svolgere la divisione euclidea (col resto) fra polinomi in una variabile.*

Esercizio 7 *Scrivere una funzione che, dato un polinomio, ne scriva il coefficiente più grande. Variante: . . . ne scriva il coefficiente più grande in valore assoluto.*

Esercizio 8 *Scrivere una funzione che, dati due monomi in x, y e z, dia in output quello che ha il grado maggiore in x; a parità, quello con il grado maggiore in y; a parità, in z. (Qui non si usino le istruzioni di CoCoA relative agli ordinamenti monomiali.)*

Ordinamenti monomiali. CoCoA permette di utilizzare vari possibili ordinamenti monomiali (e di definirne degli altri); i principali sono `Lex` (ordinamento lessicografico), `DegLex` (ordinamento lessicografico graduato), `DegRevLex` (ordinamento lessicografico graduato inverso), che è l'ordinamento predefinito, quello usato dal programma se non si si specifica altrimenti. L'indicazione dell'ordinamento da usare va data insieme alla definizione dell'anello base:

```
Use R::=Q[x,y,z,t,u,v,w], DegLex;
```

Inoltre si intende sempre che l'ordinamento sulle variabili è quello in cui compaiono nella definizione dell'anello. Quindi se definiamo l'anello `Q[x,y,z]` si intende che $x > y > z$, mentre se si definisce `Q[y,z,x]` si ha $y > z > x$.

Esercizio 9 *(Facoltativo) Sia dato il polinomio $F = x^3z + x^3 + y^2$. Essendo composto da tre monomi, può essere scritto in sei modi diversi. Ognuno di questi modi si può ottenere da uno degli ordinamenti predefiniti, eventualmente permutando le indeterminate? Se no, dimostrare che una data scrittura non si ottiene da nessun ordinamento.*

Fare lo stesso per $G = x^2 + xy + y^2$ e $H = y^2 - xz + xy$.

È inoltre possibile definire altri ordinamenti monomiali personalizzati. La notazione usata da CoCoA per un generico ordinamento è di rappresentarlo come matrice (lista di s vettori, ognuno di n componenti) $[u_1, \ldots, u_s]$. Siano $a = (a_1, \ldots, a_n)$ e $b = (b_1, \ldots, b_n)$ due n-ple di esponenti, corrispondenti cioè ai monomi $x_1^{a_1} \ldots x_n^{a_n}$ etc. Allora nel nuovo ordinamento a è maggiore di b se e solo se l's-pla $(a \cdot u_1, \ldots, a \cdot u_s)$ è maggiore lessicograficamente di $(b \cdot u_1, \ldots, b \cdot u_s)$. Qui $\cdot$ denota il prodotto scalre.

Quindi qualsiasi ordinamento è definito a partire da quello lessicografico e da un'opportuna matrice. È immediato osservare che la matrice dell'ordinamento lessicografico stesso, in questa notazione, è la matrice identità. Può essere utile convincersi del perché la matrice corrsipondente a `DegRevLex` è `[ [1, 1, 1], [0, 0, -1], [0, -1, 0] ]`. La matrice corrispondente all'ordinamento in corso può essere visualizzata con `Ord()`.

Esercizio 10 *(Facoltativo) Trovare una matrice a coefficienti non negativi che definisca* `DegRevLex`.

Esercizio 11 *(Facoltativo) Scivere una funzione che, data in input una matrice relativa a un ordinamento, ne calcoli una che descrive lo stesso ordinamento e in cui non compaiono coefficienti negativi.*

Gli operatori `LT` ("leading term"), `LC` ("leading coefficient") e `LM` ("leading monomial") sono determinati in base all'ordinamento definito. Notate che la convenzione di CoCoA è tale che il leading term moltiplicato per il leading coefficient è uguale al leading monomial (diversamente dalla più comune convenzione, in cui i ruoli del leading term e del leading monomial sono scambiati).

```
Use R::=Q[x,y,z], Lex;
P:=5x^2yz+xy^3z+xy^4;
LT(P);
x^2yz
------------------------------
LM(P);
5x^2yz
------------------------------
LC(P);
5
------------------------------
Use R::=Q[x,y,z,t], DegRevLex;
LT(x^2z + y^3z);
y^3z
------------------------------
```

Esercizio 12 *Trovare uno stesso polinomio che abbia "leading term" diverso per ognuno dei tre ordinamenti Lex, DegLex e DegRevLex.*

Ideali. Uno dei tipi di oggetti più importanti con cui CoCoA è in grado di lavorare sono gli ideali (di anelli di polinomi). Un ideale viene definito con l'istruzione `Ideal`. Per esempio,

```
I:=Ideal(x,y^2);
```

definisce I come l'ideale generato dai polinomi x e y^2, cioè $I = \langle x, y^2 \rangle$. È possibile far calcolare a CoCoA le varie operazioni tra ideali; per esempio, per ottenere l'unione e l'intersezione di due ideali I e J si usano i comandi

```
I + J;
Intersection(I,J);
```

notate che calcolare l'unione di ideali è concettualmente molto semplice, mentre l'algoritmo per trovare l'intersezione richiede più attenzione, e sarà visto in dettaglio più avanti.

La funzione `Gens` permette di ottenere un insieme di generatori per un ideale; la funzione `IsIn` dice se un elemento appartiene a un certo insieme (ideale, lista etc.):

```
x-y^3 IsIn I;
x^3-y IsIn I;
```

Basi di Groebner. Le funzioni più potenti di CoCoA fanno uso del calcolo delle basi di Groebner nel senso che, anche per molti calcoli in cui non viene chiesto esplicitamente di trovare una tale base per un ideale, CoCoA la calcola e la usa "dietro le quinte" per ottenere i risultati richiesti. Questo è per esempio il caso di `GCD`, `LCM` e divisioni tra polinomi, dell'eliminazione di una o più variabili tra i generatori di un ideale, dell'intersezione tra ideali etc. (vedi oltre).

La funzione `GBasis` permette di ottenere una base di Groebner di un ideale:

```
Use R::=Q[x,y,z];
GBasis(Ideal(xz-y^2,x^3-z^2));
[-y^2 + xz, x^3 - z^2]
------------------------------
Use R::=Q[x,y,z], Lex;
GBasis(Ideal(xz-y^2,x^3-z^2));
[xz - y^2, x^3 - z^2, -x^2y^2 + z^3, -xy^4 + z^4, -y^6 + z^5]
```

La funzione `ReducedGBasis`, ovviamente, dà la base di Groebner ridotta dell'ideale a cui si applica.

Esercizio 13 *Usando la funzione* `ReducedGBasis`*, risolvere il sistema di equazioni lineari*

$$\begin{cases} x - y + z = 2 \\ 3x - z = -6 \\ x + y = 1 \end{cases}$$

Esercizio 14 *Scrivere una funzione che calcola l'S-polinomio di due polinomi dati e, usandola, scrivere una funzione che calcola una base di Groebner di un ideale assegnato con l'algoritmo di Buchberger (senza ovviamente usare la funzione predefinita* `GBasis`*).*

Esercizio 15 *Scrivere una funzione che, data una base di Groebner, verifica se è minimale e se non lo è la rende tale.*

Esercizio 16 *Scrivere una funzione che, data una base di Groebner minimale, verifica se è ridotta e se non lo è la rende tale (anche qui, naturalmente, senza usare* `ReducedGBasis`*).*

Esercizio 17 *Usando la funzione predefinita* `GBasis` *o una propria versione, e una funzione per la divisione tra polinomi in più variabili, scrivere una funzione che dati un polinomio P e un ideale I dica se P appartiene a I.*

Eliminazione di variabili. CoCoA ha l'istruzione `Elim` che consente di eliminare una o più variabili dai generatori di un ideale. La sintassi di questo comando è `Elim(`*variabile/i*`,`*ideale*`)`. Qui *variabile/i* può essere una singola variabile oppure una lista di variabili. Per esempio:

```
Use R::=Q[t,x,y,z];
Elim(t,Ideal(x-t+5,y-2t-1,z+t-6));
Ideal(1/2y + z - 13/2, x + z - 1)
```

Questo equivale a dire che, esplicitando la t dalle equazioni del sistema

$$\begin{cases} x - t + 5 = 0 \\ y - 2t - 1 = 0 \\ z + t - 6 = 0 \end{cases}$$

si ottengono le equazioni $y/2 + z - 13/2$ e $x + z - 1$.

È possibile usare nella definizione di un anello variabili indicizzate con gli elementi di una lista (espresse nella forma `x[1..n]` per indicare le n variabili x_1, x_2, …, x_n). In questo modo si può eliminare simultaneamente più di una variabile. Per esempio:

```
Use R::=Q[t[1..3],x[1..4]];
I:=Ideal(t[1],t[2]+1-x[1],t[1]-t[3]+3-x[2],2t[1]+t[3]-2-x[3],
t[1]+2t[2]-3-x[4]);
Elim(t[1]..t[3],I);
Ideal(6x[1] + x[2] + x[3] - 3x[4] - 16, -1/3x[2] - 1/3x[3] + 1/3)
```

Naturalmente, come sappiamo dalla teoria, con la funzione `Elim` otteniamo lo stesso risultato che otterremmo costruendo una base di Groebner, rispetto all'ordinamento lessicografico, dell'ideale che stiamo studiando, e considerando i generatori in essa in cui non compaiono le variabili da eliminare. Riotteniamo così il primo esempio:

```
Use R::=Q[t,x,y,z];
I:=Ideal(x-t+5,y-2t-1,z+t-6);
GBasis(I);
[-t + x + 5, 1/2y + z - 13/2, x + z - 1]
```

dove gli ultimi due polinomi della base sono proprio quelli trovati sopra.

Questo ci permette in particolare di risolvere sistemi di equazioni polinomiali.

Implicitizzazione. Collegata col punto precedente è la possibilità di "implicitizzare" le equazioni parametriche di una varietà per ottenerne equazioni cartesiane. Anche qui la teoria ci dice quando è possibile, data una parametrizzazione di un insieme di punti, trovare la più piccola varietà che lo contiene (la sua chiusura di Zariski).

Con CoCoA, come sopra, eliminiamo i parametri prendendo, da una base di Groebner dell'ideale generato dai polinomi che uguagliati a 0 danno le equazioni parametriche, i polinomi in cui non compaiono i parametri; essi generano il cosiddetto "m-mo ideale di eliminazione" (se i parametri sono m). Così, se studiamo l'insieme di punti S parametrizzato da $x = uv$, $y = u^2$, $z = v^2$, otteniamo:

```
Use R::=Q[u,v,x,y,z], Lex;
I:=Ideal(uv-x, u^2-y, v^2-z);
GBasis(I);
[v^2 - z, u^2 - y, uv - x, -uz + vx, x^2 - yz, ux - vy]
```

Quindi il secondo ideale di eliminazione (quello generato dai polinomi in cui non compaiono u e v) è $I_2 = \langle x^2 - yz \rangle$. Ciò significa che $\mathbf{V}(x^2 - yz)$ è la più piccola varietà contenente S; sui complessi coincidono, mentre sui reali S è "metà" di $\mathbf{V}(x^2 - yz)$.

Per parametrizzazioni date in termini di funzioni razionali la teoria ci insegna che bisogna prestare ulteriore attenzione: per evitare che i denominatori delle funzioni razionali si annullino, occorre introdurre una variabile aggiuntiva e l'equazione $gw = 1$, dove g è il prodotto dei denominatori e w è la variabile aggiuntiva. In questo modo, senza introdurre indebitamente vincoli ulteriori tra le variabili, ci si limita a escludere che avvenga $g = 0$. Se per esempio vogliamo studiare la parametrizzazione $x = u^2/v$, $y = v^2/u$, $z = u$ con CoCoA, possiamo scrivere:

```
Use R::=Q[w,u,v,x,y,z], Lex;
I:=Ideal(vx-u^2,uy-v^2,u-z,1-uvw);
GBasis(I);
[u - z, v^2 - yz, vx - z^2, -wz^3 + x, -wyz^2 + v, vz - xy,
x^2y - z^3, wxy - 1]
```

ottenendo così $I_3 = \langle x^2y - z^3 \rangle$. Si provi a vedere che cosa succede senza l'equazione $1 - uvw$ che abbiamo aggiunto.

Matrice di Sylvester e risultanti. La matrice di Sylvester di due polinomi F e G rispetto alla variabile x si può ottenere in CoCoA con la funzione `Sylvester()`. Il risultante di F e G rispetto a x, cioè il determinante della relativa matrice di Sylvester, può essere ottenuto direttamente con `Resultant()`. Per esempio:

```
F:=x^2y-3xy^2+x^2-3xy;
G:=x^3y+x^3-4y^2-3y+1;
M:=Sylvester(F,G,x);
M;
Mat([
  [y + 1, -3y^2 - 3y, 0, 0, 0],
  [0, y + 1, -3y^2 - 3y, 0, 0],
  [0, 0, y + 1, -3y^2 - 3y, 0],
  [y + 1, 0, 0, -4y^2 - 3y + 1, 0],
  [0, y + 1, 0, 0, -4y^2 - 3y + 1]
])
```

```
------------------------------
D:=Det(M);
D;
-108y^9 - 513y^8 - 929y^7 - 738y^6 - 149y^5 + 112y^4 + 37y^3
- 14y^2 - 3y + 1
------------------------------
D=Resultant(F,G,x);
TRUE
------------------------------
Resultant(F,G,y);
0
```

Si osservi che CoCoA esprime una matrice come una lista di liste. L'ultima risposta, il fatto cioè che $Res(F, G, y) = 0$, ci dice che F e G hanno un fattore comune con grado positivo in y. Possiamo infatti verificare che:

```
GCD(F,G);
y + 1
```

Intersezioni e quozienti di ideali. CoCoA ha l'istruzione `Intersection` che calcola l'intersezione di due ideali. Il quoziente di due ideali I e J (cioè l'insieme dei polinomi f tali che fg sia in I per ogni g in J) si ottiene con `I:J`.

Naturalmente si può fare a meno di usare queste istruzioni:

1. calcolando i quozienti per mezzo di intersezioni (ricordando che mediante intersezioni ci si può ridurre a considerare quozienti della forma $I : \langle g \rangle$ che a loro volta sono generati da $\{h_1/g, \ldots, h_p/g\}$ se $\{h_1, \ldots, h_p\}$ è un insieme di generatori di $I \cap \langle g \rangle$);
2. calcolando l'intersezione di due ideali $I = \langle f_1, \ldots, f_r \rangle$ e $J = \langle g_1, \ldots, g_s \rangle$ mediante l'algoritmo che permette di calcolarlo in termini dei loro generatori, eliminando la variabile t dall'ideale
$$\langle tf_1, \ldots, tf_r, (1-t)g_1, \ldots, (1-t)g_s \rangle;$$
3. eliminando variabili considerando, in una base di Groebner, gli elementi in cui tali variabili non compaiono;
4. costruendo basi di Groebner esplicitamente, ad esempio con l'algoritmo di Buchberger, che fa uso essenzialmente solo di divisioni tra polinomi e ideali;
5. dividendo polinomi con l'opportuno algoritmo.

Quindi con un po' di pazienza si può riscrivere da zero parte di CoCoA...

Esercizio 18 *Come esempio di utilizzo della nozione di quoziente si può studiare la varietà* $\mathbf{V}(xz - y^2, x^3 - yz)$. *Essa contiene l'asse* z, *cioè la varietà* $\mathbf{V}(x, y)$; *per trovare l'altra componente, si considera l'ideale quoziente* $\langle xz - y^2, x^3 - yz \rangle : \langle x, y \rangle$ *e così via.*

Radicali. In CoCoA è possibile calcolare il radicale di un ideale con l'istruzione `Radical`. Per esempio:

```
Use R ::= Q[x,y];
I := Ideal(x,y)^3;
Radical(I);
Ideal(y, x)
```

Testi di riferimento. Per i concetti teorici a cui qui si fa fugace riferimento, si veda David Cox, John Little, Donal O'Shea, *Ideals, Varieties, and Algorithms*, Springer. Per i dettagli su CoCoA, si veda ovviamente il manuale, reperibile in rete.

Ultima revisione: 15 dicembre 2010

```
------------------ ----------------------------------------
------   )))   ----  CoCoA 3.7
--   ___(((_ _   --
--  |       | )  --  by L. Robbiano, A. Capani, G. Niesi,
--  | CoCoA |/   --  J. Abbott, A. Bigatti, M. Caboara,
--  |       |   ---  M. Kreuzer, D. Perkinson
---  \_____/  -----  online help   : type "Man();"
-----       -------  release notes : type "RelNotes();"
------------------ ----------------------------------------

-- Current ring is R = Q[t,x,y,z]
-------------------------------
```

```
-------------------------------------------------------
---           ___/       ___/         \             ---
--           /     _ \  /     _ \    , \             --
--           \    |   | \    |   |  ___ \            --
---          ____, __/  ____, __/ _/    _\          ---
-------------------------------------------------------
--     Version      : 4.7.4                          --
--     Online Help  : type  ?   or   ?keyword        --
--     Web site     : http://cocoa.dima.unige.it     --
-------------------------------------------------------

-------------------------------
-- The current ring is R ::= QQ[x,y,z];
-------------------------------
```