PATTERNS IN WORDS

Bruno Salvy
(Version of February 7, 1997)

This worksheet applies combstruct and gfun to a simple combinatorial model of a problem from computational biology and the study of DNA sequences. The DNA can be viewed as a long text on an alphabet of four letters (A,C,G,T). Large fragments of this text are tabulated. In particular, there are huge bases of genes, a gene being a few thousand letters long. Given a short word, it is interesting to determine whether its number of occurrences in a gene (or a virus) is far away from the most probable number of occurrences. If this number of occurrences is very improbable, then this particular word may have a biological function.

The combinatorial model is rational. The text is a word on the alphabet (a,b,c,d) where all words of the same length are equiprobable. The probability that a pattern occurs k times in the text depends on the way the pattern overlaps with itself.

```
> libname:=`/net/blagny/algo/maple/5.4`,libname:
> with(combstruct): with(gfun):
```

Specification and univariate generating functions for the pattern abab

Working over the alphabet (a,b,c,d), we first concentrate on a specific pattern (abab). To attack problems related to occurrences of this pattern in words using combstruct, we first write a *grammar* which describes a corresponding automaton.This grammar recognizes all the words on (a,b,c,d). It is written in such a way that a mark (named Mark) is present in a word everytime the pattern abab occurs. Then counting the number of marks in a word gives the number of occurences of abab in it.

```
> G:={w=Union(Epsilon,Prod(a,wa),Prod(b,w),
            Prod(c,w),Prod(d,w)),
      wa=Union(Epsilon,Prod(a,wa),Prod(b,wab),
            Prod(c,w),Prod(d,w)),
      wab=Union(Epsilon,Prod(a,waba),Prod(b,w),
            Prod(c,w),Prod(d,w)),
      waba=Union(Epsilon,Prod(a,wa),Prod(b,Prod(Mark,w)),
            Prod(c,w),Prod(d,w)),
      Mark=Epsilon,a=Atom,b=Atom,c=Atom,d=Atom}:
```

We use the function combstruct[count] to check that the number of words of length n is 4^n:

```
> count([w,G,unlabelled],size=10),4^10;
```

$$1048576,\ 1048576$$

```
> count([w,G,unlabelled],size=20),4^20;
```

$$1099511627776,\ 1099511627776$$

It is also possible to *prove* this by computing the generating function of the language with combstruct[gfsolve]:

```
> gfsolve(G,unlabelled,z);
```

$$\{\mathrm{wa}(z)=-\frac{1}{4z-1},\ \mathrm{w}(z)=-\frac{1}{4z-1},\ \mathrm{waba}(z)=-\frac{1}{4z-1},\ \mathrm{wab}(z)=-\frac{1}{4z-1},\ \mathrm{a}(z)=z,$$
$$\mathrm{b}(z)=z,\ \mathrm{Mark}(z)=1,\ \mathrm{c}(z)=z,\ \mathrm{d}(z)=z\}$$

```
> subs(",w(z));
```

$$-\frac{1}{4z-1}$$

The coefficient of z^n in the Taylor expansion of this generating function is the number 4^n of words of length n in the language, which confirms the correctness of our grammar.

Here are a few typical words of the language obtained by the *uniform random generator* provided by combstruct[draw]:

```
> to 20 do eval(subs(Prod=proc() args
  end,draw([w,G,unlabelled],size=30))) od;
```

$b, d, b, d, c, a, d, c, d, c, d, d, d, d, d, d, b, a, a, d, a, c, d, c, b, b, c, b, b, c, \mathrm{E}$

$a, c, d, c, a, b, b, c, c, b, d, a, d, a, c, a, d, a, c, a, b, b, c, a, b, a, d, a, c, d, \mathrm{E}$

$c, d, c, b, c, b, d, d, b, a, a, d, d, d, b, d, a, b, c, c, d, d, a, c, d, c, c, a, c, c, \mathrm{E}$

$c, a, c, d, b, c, d, d, c, c, a, a, c, a, a, d, a, b, d, a, b, c, d, c, d, d, a, c, c, a, \mathrm{E}$

$c, b, a, b, b, a, d, a, a, b, b, d, b, c, d, b, d, a, d, b, a, d, c, b, b, b, a, c, a, b, \mathrm{E}$

$d, d, a, c, b, c, a, b, d, b, b, a, c, a, c, a, d, a, a, b, d, a, b, c, a, d, c, d, c, d, \mathrm{E}$

$d, d, d, b, a, a, a, d, b, a, c, d, c, a, b, a, a, a, b, b, b, d, a, c, a, d, a, d, a, a, \mathrm{E}$

$b, a, c, d, b, b, c, b, d, b, a, c, a, a, c, d, b, a, b, d, d, a, c, d, c, c, c, c, b, b, \mathrm{E}$

$a, b, c, b, d, c, c, b, d, b, d, b, b, c, b, b, d, b, b, a, b, b, d, a, d, c, d, a, b, b, \mathrm{E}$

$a, d, a, a, c, d, c, c, d, d, d, d, c, c, d, c, c, b, b, b, c, d, c, b, a, a, c, b, b, b, \mathrm{E}$

$b, d, a, b, b, b, c, b, c, d, d, a, c, c, b, b, c, b, d, a, a, a, a, a, c, b, b, d, d, c, \mathrm{E}$

$b, d, c, b, c, b, c, c, c, b, c, a, b, b, c, d, d, b, a, d, c, c, d, b, c, d, c, c, d, b, \mathrm{E}$

$c, b, b, c, a, c, a, c, c, c, c, c, a, b, b, b, d, a, b, c, b, a, a, a, a, a, d, b, b, b, \mathrm{E}$

$c, a, a, a, b, d, a, b, c, c, b, b, d, a, a, c, c, b, b, c, a, b, d, b, d, d, a, a, b, c, \mathrm{E}$

$b, b, d, d, b, c, a, c, c, d, c, c, d, a, a, c, a, c, b, b, c, a, a, c, a, a, d, d, b, c, \mathrm{E}$

$b, d, d, c, a, b, a, d, b, c, a, d, a, c, d, d, a, d, b, a, b, c, a, d, b, c, d, c, c, b, \mathrm{E}$

$b, c, c, b, b, b, d, c, c, d, a, d, b, a, a, c, d, a, d, a, a, d, b, d, c, d, b, a, a, d, \mathrm{E}$

$b, b, c, b, d, b, b, b, a, a, b, a, d, c, c, b, b, c, a, a, c, d, d, a, c, c, d, b, a, b, \mathrm{E}$

$c, a, a, a, c, d, d, b, a, a, d, c, d, d, d, b, b, b, b, b, b, b, d, a, c, c, b, a, b, a, \mathrm{E}$

$a, a, a, b, a, c, c, a, d, c, d, b, b, c, d, a, c, a, a, c, b, b, b, d, a, a, c, d, b, d, \mathrm{E}$

Some of these words countain the pattern abab, as indicated by the letter 'Mark' right after its occurrence.

Bivariate generating functions

From the grammar specification above, the command combstruct[gfsolve] can also be used to derive *multivariate* generating functions. From this, it is easy to compute the probability that the pattern occurs k times in a random word of length n, the expectation of the number of occurrences of the pattern in such a word, and the corresponding variance.

```
> gfsolve(G,unlabelled,z,[[u,Mark]]);
```

$$\{\mathrm{Mark}(z,u)=u,\ \mathrm{c}(z,u)=z,\ \mathrm{d}(z,u)=z,\ \mathrm{a}(z,u)=z,\ \mathrm{b}(z,u)=z,\ \mathrm{w}(z,u)=-\frac{1+z^2}{\%1},$$

$$\mathrm{waba}(z,u)=-\frac{1-z+z^2-z^3+z\,u+z^3\,u}{\%1},\ \mathrm{wab}(z,u)=-\frac{1+z^2\,u}{\%1},$$

$$\mathrm{wa}(z,u) = -\frac{1+z^2-z^3+z^3u}{\%1}\}$$

$$\%1 := -1+4z-z^2+4z^3-z^4+z^4u$$

```
> sol:=subs(",w(z,u));
```

$$sol := -\frac{1+z^2}{-1+4z-z^2+4z^3-z^4+z^4u}$$

The coefficient of $z^n u^k$ in the Taylor series of this expression at $z=0$ is the number of words of length n where the pattern abab occurs k times. Replacing z by $\frac{z}{4}$ directly yields the *probability generating function* under the uniform model (see below for a biased model):

```
> GF:=normal(subs(z=z/4,sol));
```

$$GF := -16\frac{16+z^2}{-256+256z-16z^2+16z^3-z^4+z^4u}$$

Here are the first coefficients:

```
> S:=map(normal,series(GF,u));
```

$$S := 16\frac{16+z^2}{\%1}+16\frac{(16+z^2)z^4}{\%1^2}u+16\frac{(16+z^2)z^8}{\%1^3}u^2+16\frac{(16+z^2)z^{12}}{\%1^4}u^3+$$
$$16\frac{(16+z^2)z^{16}}{\%1^5}u^4+16\frac{(16+z^2)z^{20}}{\%1^6}u^5+O(u^6)$$

$$\%1 := 256-256z+16z^2-16z^3+z^4$$

For instance, the coefficient of u^0 in this series gives the probabilities that the pattern abab does not occur:

```
> series(coeff(S,u,0),z,31);
```

$$1+z+z^2+z^3+\frac{255}{256}z^4+\frac{127}{128}z^5+\frac{4049}{4096}z^6+\frac{2017}{2048}z^7+\frac{4019}{4096}z^8+\frac{64065}{65536}z^9+\frac{1021231}{1048576}z^{10}+$$
$$\frac{254359}{262144}z^{11}+\frac{16218481}{16777216}z^{12}+\frac{16158211}{16777216}z^{13}+\frac{16098165}{16777216}z^{14}+\frac{128306737}{134217728}z^{15}+\frac{4090557839}{4294967296}$$
$$z^{16}+\frac{2037678397}{2147483648}z^{17}+\frac{64963395809}{68719476736}z^{18}+\frac{16180495845}{17179869184}z^{19}+\frac{64481468071}{68719476736}z^{20}+$$
$$\frac{1027869544771}{1099511627776}z^{21}+\frac{16384797565471}{17592186044416}z^{22}+\frac{2040488690807}{2199023255552}z^{23}+\frac{260211964085025}{281474976710656}z^{24}$$
$$+\frac{259244982579269}{281474976710656}z^{25}+\frac{258281594502569}{281474976710656}z^{26}+\frac{1029287146005101}{1125899906842624}z^{27}+$$
$$\frac{65629579589468639}{72057594037927936}z^{28}+\frac{32692845767455035}{36028797018963968}z^{29}+\frac{1042283356801377841}{1152921504606846976}z^{30}+O(z^{31})$$

Thus the random draws of words of length 30 that we did before are typical: the probability that the pattern does not occur in such a word being

```
> coeff(",z,30)=evalf(coeff(",z,30));
```

$$\frac{1042283356801377841}{1152921504606846976} = .9040367038$$

The expected number of occurrences is obtained very directly from the bivariate generating function GF. Here is its generating function

```
> mom1:=factor(subs(u=1,diff(GF,u)));
```

$$mom1 := \frac{1}{16}\,\frac{z^4}{(z-1)^2\,(16+z^2)}$$

```
> smom1:=series(",z,31);
```

$$smom1 := \frac{1}{256}z^4 + \frac{1}{128}z^5 + \frac{47}{4096}z^6 + \frac{31}{2048}z^7 + \frac{1233}{65536}z^8 + \frac{737}{32768}z^9 + \frac{27439}{1048576}z^{10} + \frac{15647}{524288}z^{11} + \frac{562385}{16777216}z^{12} + \frac{312033}{8388608}z^{13} + \frac{10971951}{268435456}z^{14} + \frac{5979423}{134217728}z^{15} + \frac{207131857}{4294967296}z^{16} + \frac{111461089}{2147483648}z^{17} + \frac{3819399983}{68719476736}z^{18} + \frac{2036022559}{34359738368}z^{19} + \frac{69195044049}{1099511627776}z^{20} + \frac{36618683105}{549755813888}z^{21} + \frac{1236475013935}{17592186044416}z^{22} + \frac{650576084255}{8796093022208}z^{23} + \frac{21853269169361}{281474976710656}z^{24} + \frac{11444051821281}{140737488355328}z^{25} + \frac{382767009852207}{4503599627370496}z^{26} + \frac{199662180711711}{2251799813685248}z^{27} + \frac{6654107407914193}{72057594037927936}z^{28} + \frac{3459512516526817}{36028797018963968}z^{29} + \frac{114943082531089199}{1152921504606846976}z^{30} + O(z^{31})$$

```
> evalf(");
```

$$.003906250000\,z^4 + .007812500000\,z^5 + .01147460938\,z^6 + .01513671875\,z^7 + .01881408691\,z^8 + .02249145508\,z^9 + .02616786957\,z^{10} + .02984428406\,z^{11} + .03352075815\,z^{12} + .03719723225\,z^{13} + .04087370262\,z^{14} + .04455017298\,z^{15} + .04822664359\,z^{16} + .05190311419\,z^{17} + .05557958478\,z^{18} + .05925605536\,z^{19} + .06293252595\,z^{20} + .06660899654\,z^{21} + .07028546713\,z^{22} + .07396193772\,z^{23} + .07763840830\,z^{24} + .08131487889\,z^{25} + .08499134948\,z^{26} + .08866782007\,z^{27} + .09234429066\,z^{28} + .09602076125\,z^{29} + .09969723183\,z^{30} + O(z^{31})$$

Thus in a sequence of 20 draws as above, we can expect the following number of occurrences of the pattern:

```
> 20*coeff(",z,30);
```

$$1.993944637$$

The variances are computed as easily as the expectations:

```
> mom2:=factor(subs(u=1,diff(u*diff(GF,u),u)));
```

$$mom2 := -\frac{1}{128}\,\frac{z^4\,(128-128\,z+8\,z^2-8\,z^3+z^4)}{(z-1)^3\,(16+z^2)^2}$$

```
> evalf(series(mom2-add(coeff(smom1,z,i)^2*z^i,i=0..30),z,31));
```

$$.003890991211\,z^4 + .007751464844\,z^5 + .01134294271\,z^6 + .01490759850\,z^7 + .01849063463\,z^8 + .02207714226\,z^9 + .02566240294\,z^{10} + .02924733446\,z^{11} + .03283235004\,z^{12} + .03641739321\,z^{13} + .04000243074\,z^{14} + .04358746611\,z^{15} + .04717250186\,z^{16} + .05075753777\,z^{17} + .05434257365\,z^{18} + .05792760952\,z^{19} + .06151264540\,z^{20} + .06509768127\,z^{21} + .06868271715\,z^{22} + .07226775302\,z^{23} + .07585278889\,z^{24} + .07943782477\,z^{25} + .08302286064\,z^{26} + .08660789652\,z^{27} + .09019293239\,z^{28} + .09377796827\,z^{29} + .09736300414\,z^{30} + O(z^{31})$$

The coefficient of z^n in this series is the variance of the number of occurrences of the pattern in a word of length n.

Fixed number of occurrences

We now consider the probabilities that abab occurs k times, for $k = 0 .. 5$. Using gfun, we can compute these probabilities for random words of length n, with n up to a few thousands.

```
> maxnb:=5:
> for i from 0 to maxnb do proba[i]:=coeff(S,u,i) od;
```

$$proba_0 := 16 \frac{16+z^2}{256-256z+16z^2-16z^3+z^4}$$

$$proba_1 := 16 \frac{(16+z^2)\,z^4}{(256-256z+16z^2-16z^3+z^4)^2}$$

$$proba_2 := 16 \frac{(16+z^2)\,z^8}{(256-256z+16z^2-16z^3+z^4)^3}$$

$$proba_3 := 16 \frac{(16+z^2)\,z^{12}}{(256-256z+16z^2-16z^3+z^4)^4}$$

$$proba_4 := 16 \frac{(16+z^2)\,z^{16}}{(256-256z+16z^2-16z^3+z^4)^5}$$

$$proba_5 := 16 \frac{(16+z^2)\,z^{20}}{(256-256z+16z^2-16z^3+z^4)^6}$$

Since we want to investigate these probabilities for texts of large size (a typical gene is a few thousand letters long), we need the Taylor expansions of these rational functions for very large orders. These can be computed using gfun, which will first compute the *linear* recurrence satisfied by these Taylor coefficients (gfun[diffeqtorec]), and then exploit these recurrences to compute the series efficiently (gfun[rectoproc]):

```
> for i from 0 to maxnb do
     rec:=diffeqtorec(y(z)-proba[i],y(z),u(n));
     print(i,rec);
     rec:=select(has,rec,n) union
  {seq(op(1,i)=evalf(op(2,i)),i=remove(has,rec,n))};
     P[i]:=rectoproc(rec,u(n),remember)
  od:
```

$$0, \{u(0)=1, u(1)=1, u(2)=1, u(3)=1,\\ u(n)-16\,u(n+1)+16\,u(n+2)-256\,u(n+3)+256\,u(n+4)\}$$

$$1, \{u(n)-32\,u(n+1)+288\,u(n+2)-1024\,u(n+3)+8960\,u(n+4)-16384\,u(n+5)\\ +73728\,u(n+6)-131072\,u(n+7)+65536\,u(n+8), u(4)=\frac{1}{256}, u(5)=\frac{1}{128},\\ u(6)=\frac{47}{4096}, u(7)=\frac{31}{2048}, u(3)=0, u(0)=0, u(1)=0, u(2)=0\}$$

$$2, \{u(n)-48\,u(n+1)+816\,u(n+2)-6400\,u(n+3)+38400\,u(n+4)\\ -258048\,u(n+5)+815104\,u(n+6)-4128768\,u(n+7)+9830400\,u(n+8)\\ -26214400\,u(n+9)+53477376\,u(n+10)-50331648\,u(n+11)$$

$+ 16777216\, u(n+12),\ u(3)=0,\ u(0)=0,\ u(1)=0,\ u(2)=0,\ u(8)=\frac{1}{65536},$

$u(9)=\frac{3}{65536},\ u(10)=\frac{47}{524288},\ u(11)=\frac{77}{524288},\ u(5)=0,\ u(7)=0,\ u(6)=0,\ u(4)=0\}$

$3, \{u(n) - 64\, u(n+1) + 1600\, u(n+2) - 20480\, u(n+3) + 166400\, u(n+4)$

$- 1196032\, u(n+5) + 7405568\, u(n+6) - 32768000\, u(n+7) + 164823040\, u(n+8)$

$- 524288000\, u(n+9) + 1895825408\, u(n+10) - 4898947072\, u(n+11)$

$+ 10905190400\, u(n+12) - 21474836480\, u(n+13) + 26843545600\, u(n+14)$

$- 17179869184\, u(n+15) + 4294967296\, u(n+16),\ u(8)=0,\ u(9)=0,\ u(10)=0,$

$u(11)=0,\ u(3)=0,\ u(0)=0,\ u(1)=0,\ u(2)=0,\ u(12)=\frac{1}{16777216},\ u(13)=\frac{1}{4194304},$

$u(14)=\frac{157}{268435456},\ u(15)=\frac{77}{67108864},\ u(5)=0,\ u(7)=0,\ u(6)=0,\ u(4)=0\}$

$4, \{u(17)=\frac{5}{4294967296},\ u(18)=\frac{59}{17179869184},\ u(n) - 80\, u(n+1) + 2640\, u(n+2)$

$- 47360\, u(n+3) + 536320\, u(n+4) - 4612096\, u(n+5) + 34856960\, u(n+6)$

$- 218234880\, u(n+7) + 1156382720\, u(n+8) - 5887754240\, u(n+9)$

$+ 23206035456\, u(n+10) - 94204067840\, u(n+11) + 296033976320\, u(n+12)$

$- 893890068480\, u(n+13) + 2284385730560\, u(n+14) - 4836133175296\, u(n+15)$

$+ 8997956485120\, u(n+16) - 12713103196160\, u(n+17) + 11338713661440\, u(n+18)$

$- 5497558138880\, u(n+19) + 1099511627776\, u(n+20),\ u(8)=0,\ u(9)=0,\ u(10)=0,$

$u(11)=0,\ u(3)=0,\ u(0)=0,\ u(1)=0,\ u(2)=0,\ u(12)=0,\ u(14)=0,\ u(15)=0,$

$u(13)=0,\ u(19)=\frac{135}{17179869184},\ u(16)=\frac{1}{4294967296},\ u(5)=0,\ u(7)=0,\ u(6)=0,$

$u(4)=0\}$

$5, \{u(16)=0,\ u(17)=0,\ u(18)=0,\ u(19)=0,\ -6262818231812096\, u(n+21)$

$+ 4327677766926336\, u(n+22) - 1688849860263936\, u(n+23)$

$+ 281474976710656\, u(n+24) + 427546814447616\, u(n+16)$

$- 14585708937216\, u(n+13) + 51380156891136\, u(n+14)$

$- 155589485264896\, u(n+15) - 999161856\, u(n+7) + 6523846656\, u(n+8)$

$- 14647296\, u(n+5) + 130031616\, u(n+6) + 5828511138840576\, u(n+20)$

$- 1047697142317056\, u(n+17) + 2181568508461056\, u(n+18)$

$- 3931853580926976\, u(n+19) + u(n) - 96\, u(n+1) + 3936\, u(n+2) - 91136\, u(n+3)$

$+ 1357056\, u(n+4) + 3977894690816\, u(n+12) - 911606808576\, u(n+11)$

$+ 200703737856\, u(n+10) - 37985714176\, u(n+9),\ u(8)=0,\ u(9)=0,\ u(10)=0,$

$u(11)=0,\ u(3)=0,\ u(0)=0,\ u(1)=0,\ u(2)=0,\ u(12)=0,\ u(14)=0,\ u(15)=0,$

$u(13)=0,\ u(5)=0,\ u(7)=0,\ u(6)=0,\ u(4)=0,\ u(20)=\frac{1}{1099511627776},$

$u(21)=\frac{3}{549755813888},\ u(22)=\frac{331}{17592186044416},\ u(23)=\frac{433}{8796093022208}\}$

We now compute the probability that the pattern abab occurs 0,1,...,5 times in a word of length 1000:

```
> Digits:=30:for i from 0 to maxnb do i,P[i](1000) od;
```

0, .0244226133931942053951590959829

1, .0918167243004940139730071586260

2, .171416402175840013700533101824

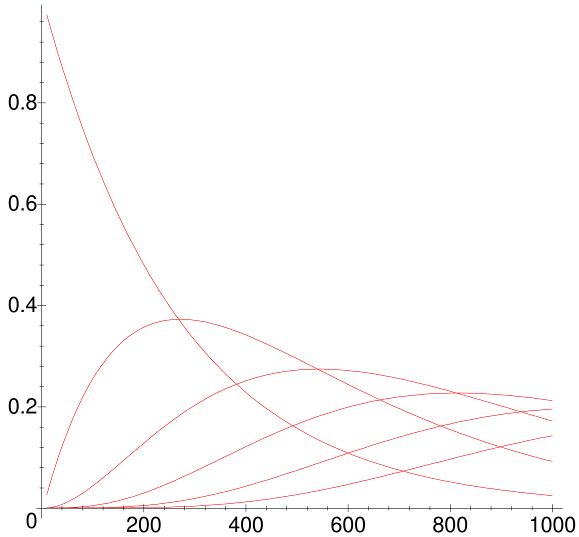
3, .211888305461657610548046662164

4, .195084324568944893971277345250

5, .142695853051912317834459204078

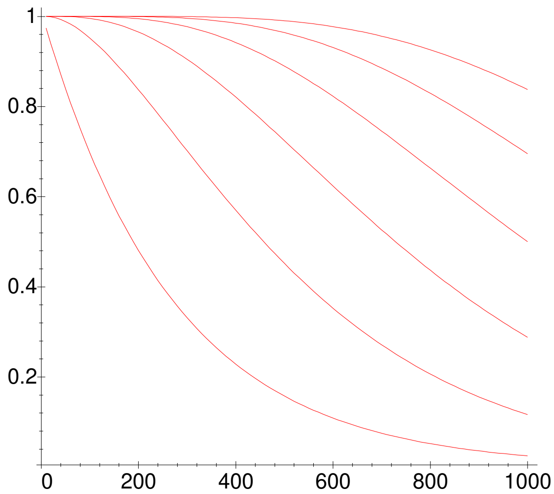
The following picture then shows how these probabilities evolve with the length of the word:

```
> plots[display]({seq(plot([seq([10*i,P[j](10*i)],i=1..100)]),j
  =0..maxnb)});
```



Similarly, the following picture indicates the probabilities that the pattern abab occurs at most k times instead of exactly k times

```
> plots[display]({seq(plot([seq([10*i,add(P[k](10*i),k=0..j)],i
  =1..100)]),j=0..maxnb)});
```



The complexity of these computations grows only linearly with the number k of occurrences under study. Other kinds of constraints like number of occurrences larger than k for fixed k or between k_1 and k_2 also give rise to rational generating functions that can be extracted from the generating function GF and thus can be treated the same way.

Other patterns

All the above computation was derived from the grammar describing the language, a mark being appended to every occurrence of the pattern. It is actually easy to write a Maple procedure taking as input a word, and producing the corresponding combstruct grammar. Then the whole computation above can be reproduced for any pattern completely automatically.

```
> gengram:=proc(pattern::list({identical(a),identical(b),identical(c),identical(d)}))local i, eq, letter, state, j;
  for i to nops(pattern) do for letter in [a,b,c,d] do for j from 0 to i-1 do if
  [op(1..i-j,pattern)]=[op(j+1..i-1,pattern),letter] then
  state[letter]:=cat(w,op(1..i-j,pattern)); break fi od; if j=i
  then state[letter]:=w fi
  od;eq[i]:=cat(w,op(1..i-1,pattern))=Union(Epsilon,seq(Prod(letter,state[letter]),letter=[a,b,c,d])) od;
  subs(cat(w,op(pattern))=Prod(Mark,w),{seq(eq[i],i=1..nops(pattern)),seq(letter=Atom,letter=[a,b,c,d]),Mark=Epsilon}) end;
```

$gengram :=$ **proc**($pattern$::list({ identical(a), identical(b), identical(c), identical(d) }))

local i, eq, $letter$, $state$, j;

for i **to** nops($pattern$) **do**

for $letter$ **in** $[a, b, c, d]$ **do**

for j **from** 0 **to** $i-1$ **do**

if [op($1 .. i-j$, $pattern$)] = [op($j+1 .. i-1$, $pattern$), $letter$] **then**

$state[letter]$:= cat(w, op($1 .. i-j$, $pattern$)); $break$

fi

od;

if $j = i$ **then** $state[letter] := w$ **fi**

od;

$eq[i]$:= cat(w, op($1 .. i-1$, $pattern$)) =

Union(E, seq(Prod($letter$, $state[letter]$), $letter = [a, b, c, d]$))

od;

subs(cat(w, op($pattern$)) = Prod($Mark$, w), { $Mark$ = E,

seq($eq[i]$, $i = 1 ..$ nops($pattern$)), seq($letter = Atom$, $letter = [a, b, c, d]$) })

end

Given a pattern, this procedure outputs the corresponding combstruct grammar. Thus for instance, using the pattern abab,

```
> gengram([a,b,a,b]);
```

$\{ w = \text{Union}(\text{E}, \text{Prod}(a, wa), \text{Prod}(b, w), \text{Prod}(c, w), \text{Prod}(d, w)),$

$wa = \text{Union}(\text{E}, \text{Prod}(a, wa), \text{Prod}(b, wab), \text{Prod}(c, w), \text{Prod}(d, w)),$

$wab = \text{Union}(\text{E}, \text{Prod}(a, waba), \text{Prod}(b, w), \text{Prod}(c, w), \text{Prod}(d, w)),$

$waba = \text{Union}(\text{E}, \text{Prod}(a, wa), \text{Prod}(b, \text{Prod}(Mark, w)), \text{Prod}(c, w), \text{Prod}(d, w)),$

$Mark = \text{E}, a = Atom, b = Atom, c = Atom, d = Atom \}$

we obtain the grammar we started with. It is now possible to study longer patterns easily. Here are the different states leading to the probability that the pattern abacab occurs twice in a word of length 5000:

First the grammar is generated

```
> G:=gengram([a,b,a,c,a,b]);
```

$$G := \{ w = \mathrm{Union}(\mathrm{E}, \mathrm{Prod}(a, wa), \mathrm{Prod}(b, w), \mathrm{Prod}(c, w), \mathrm{Prod}(d, w)),$$
$$wa = \mathrm{Union}(\mathrm{E}, \mathrm{Prod}(a, wa), \mathrm{Prod}(b, wab), \mathrm{Prod}(c, w), \mathrm{Prod}(d, w)),$$
$$wab = \mathrm{Union}(\mathrm{E}, \mathrm{Prod}(a, waba), \mathrm{Prod}(b, w), \mathrm{Prod}(c, w), \mathrm{Prod}(d, w)), Mark = \mathrm{E},$$
$$a = Atom, b = Atom, c = Atom, d = Atom,$$
$$waba = \mathrm{Union}(\mathrm{E}, \mathrm{Prod}(a, wa), \mathrm{Prod}(b, wab), \mathrm{Prod}(c, wabac), \mathrm{Prod}(d, w)),$$
$$wabac = \mathrm{Union}(\mathrm{E}, \mathrm{Prod}(a, wabaca), \mathrm{Prod}(b, w), \mathrm{Prod}(c, w), \mathrm{Prod}(d, w)),$$
$$wabaca = \mathrm{Union}(\mathrm{E}, \mathrm{Prod}(a, wa), \mathrm{Prod}(b, \mathrm{Prod}(Mark, w)), \mathrm{Prod}(c, w), \mathrm{Prod}(d, w))\}$$

then the bivariate generating functions are derived

```
> gfsolve(G,unlabelled,z,[[u,Mark]]);
```

$$\{ \mathrm{Mark}(z, u) = u, \mathrm{c}(z, u) = z, \mathrm{d}(z, u) = z, \mathrm{a}(z, u) = z, \mathrm{b}(z, u) = z,$$
$$\mathrm{wabac}(z, u) = -\frac{1 + z^4 - z^2 + z^2 u}{\%1}, \mathrm{wabaca}(z, u) = -\frac{1 - z + z^4 - z^5 + z u + z^5 u}{\%1},$$
$$\mathrm{w}(z, u) = -\frac{1 + z^4}{\%1}, \mathrm{wa}(z, u) = -\frac{1 + z^4 - z^5 + z^5 u}{\%1},$$
$$\mathrm{waba}(z, u) = -\frac{-z^3 + z^3 u + 1 + z^4 - z^5 + z^5 u}{\%1}, \mathrm{wab}(z, u) = -\frac{1 + z^4 u}{\%1}\}$$
$$\%1 := -1 + 4z - z^4 + 4z^5 - z^6 + z^6 u$$

the generating function is then converted into a probability generating function by a simple substitution

```
> normal(subs(",z=z/4,w(z,u)));
```

$$-16 \frac{256 + z^4}{-4096 + 4096 z - 16 z^4 + 16 z^5 - z^6 + z^6 u}$$

extracting a coefficient we get the probability generating function of words with 2 occurrences of the pattern

```
> coeff(series(",u,3),u,2);
```

$$-16 \frac{(256 + z^4) z^{12}}{(4096 - 4096 z + 16 z^4 - 16 z^5 + z^6)^2 (-4096 + 4096 z - 16 z^4 + 16 z^5 - z^6)}$$

this gives rise to a linear recurrence satisfied by the Taylor coefficients

```
> diffeqtorec(y(z)-",y(z),u(n));
```

$$\{ \mathrm{u}(n) - 48 \mathrm{u}(n+1) + 816 \mathrm{u}(n+2) - 5632 \mathrm{u}(n+3) + 13056 \mathrm{u}(n+4) - 24576 \mathrm{u}(n+5)$$
$$+ 409600 \mathrm{u}(n+6) - 3932160 \mathrm{u}(n+7) + 9830400 \mathrm{u}(n+8) - 9437184 \mathrm{u}(n+9)$$
$$+ 53477376 \mathrm{u}(n+10) - 905969664 \mathrm{u}(n+11) + 2466250752 \mathrm{u}(n+12)$$
$$- 2415919104 \mathrm{u}(n+13) + 805306368 \mathrm{u}(n+14) - 68719476736 \mathrm{u}(n+15)$$
$$+ 206158430208 \mathrm{u}(n+16) - 206158430208 \mathrm{u}(n+17) + 68719476736 \mathrm{u}(n+18),$$
$$\mathrm{u}(8) = 0, \mathrm{u}(9) = 0, \mathrm{u}(10) = 0, \mathrm{u}(11) = 0, \mathrm{u}(3) = 0, \mathrm{u}(0) = 0, \mathrm{u}(1) = 0, \mathrm{u}(2) = 0,$$
$$\mathrm{u}(16) = \frac{1919}{2147483648}, \mathrm{u}(17) = \frac{2685}{2147483648}, \mathrm{u}(13) = \frac{3}{16777216}, \mathrm{u}(14) = \frac{3}{8388608},$$
$$\mathrm{u}(15) = \frac{5}{8388608}, \mathrm{u}(12) = \frac{1}{16777216}, \mathrm{u}(5) = 0, \mathrm{u}(7) = 0, \mathrm{u}(6) = 0, \mathrm{u}(4) = 0 \}$$

As before, in order to speed up the computation, we change the initial conditions into floating point numbers:

```
> Digits:=50:
> select(has,"",n) union
  {seq(op(1,i)=evalf(op(2,i)),i=remove(has,"",n))}:
```

gfun[rectoproc] then produces a procedure to evaluate this sequence efficiently

```
> p:=rectoproc(",u(n),remember):
> for i from 0 by 500 to 10000 do i,p(i) od;
```

0, 0

500, .0063150117243680006941737790410535189675290303636

1000, .022845669667738748880934316764827696531617920057217

1500, .045831984772510805850664512380482793949229398530993

2000, .072392544038562041881913681079140719424710753304

2500, .10035746745088688975723961476239676831030786503289

3000, .12812757778202661088983274047592265825202655995483

3500, .15455829222728339134927804787304761213688949279302

4000, .17886421404362208214923066549377196843072995657744

4500, .20054102302145639224051350423746279347093100622187

5000, .21930179214199336246264295052259709850462664724703

5500, .23502530743720237653362593359320963038122232373486

6000, .24771435028221900083612568141702821995338957743035

6500, .25746222590816597326537981877020117502313978844704

7000, .26442609723657607176020428789794004398686669674917

7500, .26880591641949851274334805190883721711332228935488

8000, .27082794390794998226423515157595905661403600129471

8500, .27073201176646921007029392735947556297190594300234

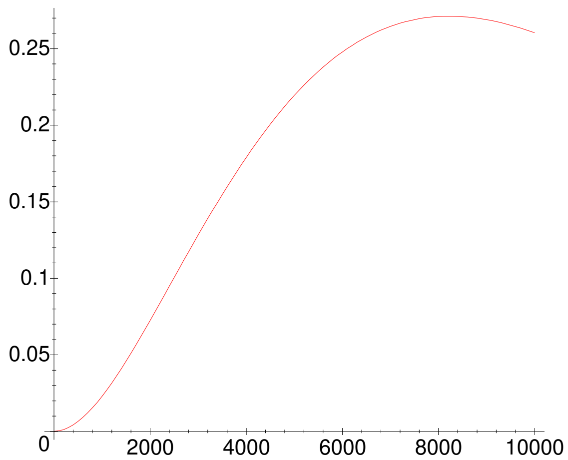
9000, .26876182884268468498278258876318825718680056130915

9500, .26515774417307687858459136638391533890066702975422

10000, .26015148498223275424933139956792026474093524442624

The following picture shows the evolution of this probability with the length of the word

```
> plot([seq([100*i,p(100*i)],i=0..100)]);
```



Extensions

Non uniform distribution

The bases A,C,G,T in DNA are not equiprobable. The combstruct model can be refined as follows to take into account such irregularities. We use here the values A=0.25, C=0.18, G=0.24, T=0.33 which come from a viral DNA (ϕX174). The pattern is the same as above. We only have to add marks in the grammar, compute a multivariate generating function and substitute the probabilities in it.

```
> G;
```

$$\{w = \mathrm{Union}(\mathrm{E}, \mathrm{Prod}(a, wa), \mathrm{Prod}(b, w), \mathrm{Prod}(c, w), \mathrm{Prod}(d, w)),$$
$$wa = \mathrm{Union}(\mathrm{E}, \mathrm{Prod}(a, wa), \mathrm{Prod}(b, wab), \mathrm{Prod}(c, w), \mathrm{Prod}(d, w)),$$
$$wab = \mathrm{Union}(\mathrm{E}, \mathrm{Prod}(a, waba), \mathrm{Prod}(b, w), \mathrm{Prod}(c, w), \mathrm{Prod}(d, w)), Mark = \mathrm{E},$$
$$a = Atom, b = Atom, c = Atom, d = Atom,$$
$$waba = \mathrm{Union}(\mathrm{E}, \mathrm{Prod}(a, wa), \mathrm{Prod}(b, wab), \mathrm{Prod}(c, wabac), \mathrm{Prod}(d, w)),$$
$$wabac = \mathrm{Union}(\mathrm{E}, \mathrm{Prod}(a, wabaca), \mathrm{Prod}(b, w), \mathrm{Prod}(c, w), \mathrm{Prod}(d, w)),$$
$$wabaca = \mathrm{Union}(\mathrm{E}, \mathrm{Prod}(a, wa), \mathrm{Prod}(b, \mathrm{Prod}(Mark, w)), \mathrm{Prod}(c, w), \mathrm{Prod}(d, w))\}$$

Marks are added by replacing a by Prod(Atom,Marka) everywhere it occurs, and similarly for the other letters. There Marka, Markb, Markc and Markd have size 0 and do not modify the counting sequences and the related probabilities, but make it possible to extract multivariate generating functions which contain more information.

```
> Gprob:=subs(a=Prod(Atom,Marka),b=Prod(Atom,Markb),c=Prod(A
  tom,Markc),d=Prod(Atom,Markd),G minus
  {a=Atom,b=Atom,c=Atom,d=Atom}) union {Marka=Epsilon,
  Markb=Epsilon, Markc=Epsilon, Markd=Epsilon};
```

$$Gprob := \{Mark = \mathrm{E}, Markd = \mathrm{E}, Marka = \mathrm{E}, Markb = \mathrm{E}, Markc = \mathrm{E}, wabac = \mathrm{Union}($$
$$\mathrm{E}, \mathrm{Prod}(\mathrm{Prod}(Atom, Marka), wabaca), \mathrm{Prod}(\mathrm{Prod}(Atom, Markb), w), \%3, \%2),$$
$$wabaca = \mathrm{Union}(\mathrm{E}, \%1, \mathrm{Prod}(\mathrm{Prod}(Atom, Markb), \mathrm{Prod}(Mark, w)), \%3, \%2),$$
$$wa = \mathrm{Union}(\mathrm{E}, \%1, \mathrm{Prod}(\mathrm{Prod}(Atom, Markb), wab), \%3, \%2),$$
$$w = \mathrm{Union}(\mathrm{E}, \%1, \mathrm{Prod}(\mathrm{Prod}(Atom, Markb), w), \%3, \%2), wab =$$
$$\mathrm{Union}(\mathrm{E}, \mathrm{Prod}(\mathrm{Prod}(Atom, Marka), waba), \mathrm{Prod}(\mathrm{Prod}(Atom, Markb), w), \%3, \%2),$$
$$waba = \mathrm{Union}(\mathrm{E}, \%1, \mathrm{Prod}(\mathrm{Prod}(Atom, Markb), wab),$$
$$\mathrm{Prod}(\mathrm{Prod}(Atom, Markc), wabac), \%2)\}$$
$$\%1 := \mathrm{Prod}(\mathrm{Prod}(Atom, Marka), wa)$$
$$\%2 := \mathrm{Prod}(\mathrm{Prod}(Atom, Markd), w)$$
$$\%3 := \mathrm{Prod}(\mathrm{Prod}(Atom, Markc), w)$$

Here is the multivariate generating function:

```
> GF:=subs(gfsolve(Gprob,unlabelled,z,[[u,Mark],[a,Marka],[b
  ,Markb],[c,Markc],[d,Markd]]),w(a,b,d,z,u,c));
```

$$GF := -(1 + z^4\, c\, b\, a^2) \Big/ (-1 + z\, c + z\, d + z\, b + z\, a - z^4\, c\, b\, a^2 + z^5\, c^2\, b\, a^2$$
$$+ z^6\, c\, a^3\, b^2\, u + z^5\, c\, b^2\, a^2 + z^5\, c\, d\, b\, a^2 - z^6\, c\, a^3\, b^2 + z^5\, c\, a^3\, b)$$

The coefficient of $a^k\, b^l\, c^m\, d^p\, u^q\, z^n$ in the Taylor expansion of GF in z is the number of words of length n with k occurrences of the letter a, l occurrences of the letter b ... and q occurrences of the pattern. The probability generating function is obtained by substituting a, b, c, d by the corresponding probability. Thus GF now takes into account both the specific pattern and the biased probabilities of the letters in the text.

We can again compute the probability that the pattern occurs twice in words of length n and compare this with the uniform model:

```
> normal(coeff(series(GF,u,3),u,2));
```

$$(1+z^4\,c\,b\,a^2)\,z^{12}\,c^2\,a^6\,b^4 \Big/ (1-z\,c-z\,d-z\,b-z\,a+z^4\,c\,b\,a^2-z^5\,c^2\,b\,a^2-z^5\,c\,a^3\,b - z^5\,c\,b^2\,a^2-z^5\,c\,d\,b\,a^2+z^6\,c\,a^3\,b^2)^3$$

Again we compute a linear recurrence satisfied by the Taylor coefficients from which we produce an efficient procedure for their evaluation:

```
> diffeqtorec(y(z)-",y(z),u(n)):
> nuprob:=rectoproc(subs([a=0.25,b=0.18,c=0.24,d=0.33],"),u(
  n),remember);
```

nuprob := **proc**(*n*)

option *remember*;

−.1793613375000000000000000*10^(-11)*procname(*n* − 18)
+ .1195742250000000000000000*10^(-9)*procname(*n* − 17)
− .27767792250000000000000*10^(-8)*procname(*n* − 16)
+ .2499741000000000000000*10^(-7)*procname(*n* − 15)
− .61706205000000000000*10^(-7)*procname(*n* − 14)
+ .1033357500000000000*10^(-6)*procname(*n* − 13)
− .2032269750000000000*10^(-5)*procname(*n* − 12)
+ .0000258066000000000000*procname(*n* − 11)
− .0000675783000000000*procname(*n* − 10)
+ .000065610000000000*procname(*n* − 9)
− .0003863700000000*procname(*n* − 8) + .00882900000000*procname(*n* − 7)
− .024664500000*procname(*n* − 6) + .0243000000*procname(*n* − 5)
− .00810000*procname(*n* − 4) + 1.000000*procname(*n* − 3)
− 3.0000*procname(*n* − 2) + 3.00*procname(*n* − 1)

end

The probabilities for this non-uniform model are rather different from those obtained in the uniform model: the pattern abacab is now less probable, since b and c are less probable than in the uniform model.

```
> for i from 0 by 500 to 10000 do i,nuprob(i) od;
```

0, 0
500, .0016611289704324688854388545958964125513943988948100
1000, .006388305398793279934673668650469528650395330187121
1500, .013623871864480822910650492450210026106793337868535
2000, .022875743659316076727562582096197415295198442816375
2500, .033711674813168131315236170908372047990342960692941
3000, .045753295836612073727005319179753446921480127987881
3500, .05867068262676322653968990525567903769965439339474
4000, .07217741251365880922714649773161309660746040896729
4500, .08602606689303329360605786600083735134000578954674
5000, .10000414309659279085114190777610195235516638340055
5500, .11393034111234219558244886102228686576931649134514
6000, .12765119350316385127900485277639943864632820741055

6500, .14103800939839625650655356639300360918969298889808

7000, .15398410576611658574837502387791534589595693505111

7500, .16640230132757062179060380465870312546530028301843

8000, .17822265046305504690061295363839222249470354002326

8500, .18939039629290880717842739803582976967022386165001

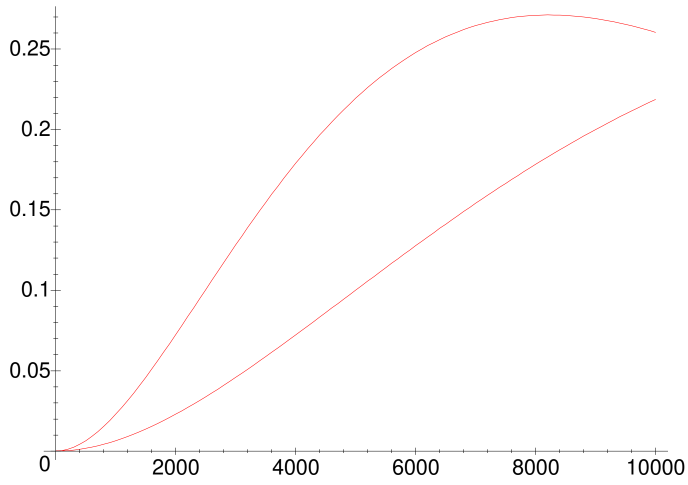
9000, .19986412380958213725772106796872549592405261523191

9500, .20961409549765834787696021705546916917884804737025

10000, .21862075331806358464328203310760351493139896029072

Here are the curves corresponding to the non-uniform and to the uniform model:

```
> plots[display]({plot([seq([100*i,p(100*i)],i=0..100)]),plo
  t([seq([100*i,nuprob(100*i)],i=0..100)])});
```



Markovian model

By slightly modifying the automaton, it is possible to consider a Markovian model, where instead of giving the probabilities of occurrence of each letter, one gives the probabilities of transition from one letter to the next one. The new automaton is almost the same as the previous one, except that the transitions are marked and three more states are added at the begining. Here is the procedure generating the new automaton from the pattern:

```
> gengram2 := proc (pattern::list({identical(a),
  identical(b),identical(c), identical(d)}))local i,
  eq, letter, state, j,alph;alph:=[a,b,c,d];for i to
  nops(pattern) do for letter in alph do for j from
  0 to i-1 do if [op(1 .. i-j,pattern)] =[op(j+1 ..
  i-1,pattern), letter] then state[letter]
  :=cat(w,op(1 .. i-j,pattern));break fi od;if j=i
  then state[letter] := cat(w,letter) fi od;eq[i] :=
  cat(w,op(1 .. i-1,pattern))
  =Union(Epsilon,seq(Prod(letter,`if`(i>1,cat(Mark,p
  attern[i-1],letter),cat(Markini,letter)),state[let
  ter]),     letter =
  alph))od;subs(cat(w,op(pattern))
  =Prod(Mark,w),{Mark =
```

```
Epsilon,seq(seq(cat(Mark,i,j)=Epsilon,j=alph),i=al
ph),seq(eq[i],i = 1 .. nops(pattern)),seq(letter =
Atom,letter=alph),
seq(cat(w,i)=Union(Epsilon,seq(Prod(j,cat(Mark,i,j
),cat(w,j)),j=alph)),i=subs(pattern[1]=NULL,alph))
,seq(cat(Markini,i)=Epsilon,i=alph)})end;
```

$gengram2$:= **proc**(
$pattern$::list({ identical(a), identical(b), identical(c), identical(d) }))
local i, eq, $letter$, $state$, j, $alph$;
  $alph$:= [a, b, c, d];
  for i **to** nops($pattern$) **do**
    for $letter$ **in** $alph$ **do**
      for j **from** 0 **to** $i-1$ **do**
        if [op(1 .. $i-j$, $pattern$)] = [op($j+1$ .. $i-1$, $pattern$), $letter$] **then**
          $state$[$letter$] := cat(w, op(1 .. $i-j$, $pattern$)); *break*
        fi
      od;
      if $j = i$ **then** $state$[$letter$] := cat(w, $letter$) **fi**
    od;
    eq[i] := cat(w, op(1 .. $i-1$, $pattern$)) = Union(E, seq(Prod($letter$,
      ‘if‘($1 < i$, cat($Mark$, $pattern$[$i-1$], $letter$), cat($Markini$, $letter$)),
      $state$[$letter$]), $letter = alph$))
  od;
  subs(cat(w, op($pattern$)) = Prod($Mark$, w), { $Mark$ = E,
    seq(cat($Markini$, i) = E, $i = alph$), seq(
    cat(w, i) = Union(E, seq(Prod(j, cat($Mark$, i, j), cat(w, j)), $j = alph$)),
    i = subs($pattern$[1] = $NULL$, $alph$)), seq($letter = Atom$, $letter = alph$),
    seq(seq(cat($Mark$, i, j) = E, $j = alph$), $i = alph$),
    seq(eq[i], i = 1 .. nops($pattern$)) })
end

For instance, the same pattern abacab as above yields the automaton described by the following grammar

```
> G:=gengram2([a,b,a,c,a,b]);
```

$$G := \{ Mark = E, a = Atom, b = Atom, c = Atom, d = Atom, w = \mathrm{Union}(E, \mathrm{Prod}(a, Markinia, wa), \mathrm{Prod}(b, Markinib, wb), \mathrm{Prod}(c, Markinic, wc), \mathrm{Prod}(d, Markinid, wd)), wabac = \mathrm{Union}(E, \mathrm{Prod}(a, Markca, wabaca), \mathrm{Prod}(b, Markcb, wb), \mathrm{Prod}(c, Markcc, wc), \mathrm{Prod}(d, Markcd, wd)), waba = \mathrm{Union}(E, \mathrm{Prod}(a, Markaa, wa), \mathrm{Prod}(b, Markab, wab), \mathrm{Prod}(c, Markac, wabac), \mathrm{Prod}(d, Markad, wd)), wab = \mathrm{Union}(E, \mathrm{Prod}(a, Markba, waba), \mathrm{Prod}(b, Markbb, wb), \mathrm{Prod}(c, Markbc, wc), \mathrm{Prod}(d, Markbd, wd)), wa = \mathrm{Union}(E, \mathrm{Prod}(a, Markaa, wa), \mathrm{Prod}(b, Markab, wab), \mathrm{Prod}(c, Markac, wc), \mathrm{Prod}(d, Markad, wd)), Markda = E, Markdc = E, Markdb = E, Markdd = E, wabaca$$

$= \mathrm{Union}(\mathrm{E}, \mathrm{Prod}(a, Markaa, wa), \mathrm{Prod}(b, Markab, \mathrm{Prod}(Mark, w)),$
$\mathrm{Prod}(c, Markac, wc), \mathrm{Prod}(d, Markad, wd)), Markbd = \mathrm{E}, Markcb = \mathrm{E}, Markcc = \mathrm{E},$
$Markcd = \mathrm{E}, Markca = \mathrm{E}, Markba = \mathrm{E}, Markbb = \mathrm{E}, Markbc = \mathrm{E}, Markaa = \mathrm{E},$
$Markac = \mathrm{E}, Markad = \mathrm{E}, Markab = \mathrm{E}, wd = \mathrm{Union}(\mathrm{E}, \mathrm{Prod}(a, Markda, wa),$
$\mathrm{Prod}(b, Markdb, wb), \mathrm{Prod}(c, Markdc, wc), \mathrm{Prod}(d, Markdd, wd)), wc = \mathrm{Union}(\mathrm{E},$
$\mathrm{Prod}(a, Markca, wa), \mathrm{Prod}(b, Markcb, wb), \mathrm{Prod}(c, Markcc, wc),$
$\mathrm{Prod}(d, Markcd, wd)), wb = \mathrm{Union}(\mathrm{E}, \mathrm{Prod}(a, Markba, wa), \mathrm{Prod}(b, Markbb, wb),$
$\mathrm{Prod}(c, Markbc, wc), \mathrm{Prod}(d, Markbd, wd)), Markinic = \mathrm{E}, Markinid = \mathrm{E},$
$Markinia = \mathrm{E}, Markinib = \mathrm{E}\}$

From there, generating functions follow. For instance, we recover the generating function obtained before when all transitions are equiprobable:

```
> subs(gfsolve(G,unlabelled,z,[[u,Mark],seq([1/4,cat(Markini
  ,i)],i=[a,b,c,d]),seq(seq([1/4,cat(Mark,i,j)],j=[a,b,c,d])
  ,i=[a,b,c,d])]),w(z,u));
```

$$-16\,\frac{256+z^4}{z^6\,u+16\,z^5-z^6-4096+4096\,z-16\,z^4}$$

The rest of the treatment is as before.

Multiple patterns

It is also possible to write an automaton which will recognize not only a fixed pattern, but a set of possible patterns. The procedure gengram3 below takes as input a list of patterns, and produces the minimal grammar recognizing all words on 4 letters, the patterns being ‘‘marked’’.

```
> gengram3:=proc
  (patterns::list(list({identical(a),identical(b),identical(
  c),identical(d)}))) local alpha, states, state, eq, n,
  trans, letter, nbst, st, indst, i; alpha:=[a,b,c,d];
  nbst:=1; states[1]:=[]; for indst while indst<=nbst do
  state:=states[indst]; n:=nops(state); for letter in alpha
  do if member([op(state),letter],patterns) then
  trans[letter]:=Prod(Mark,w) elif
  member([op(state),letter],map( proc(x,n) if nops(x)>n then
  [op(1..n+1,x)] fi end, patterns,nops(state))) then
  trans[letter]:=cat(w,op(state),letter); nbst:=nbst+1;
  states[nbst]:=[op(state),letter] else for i from indst by
  -1 to 2 do st:=states[i]; if
  st=[op(n-nops(st)+2..n,state),letter] then
  trans[letter]:=cat(w,op(st)); break fi od; if i=1 then
  trans[letter]:=w fi fi od; eq[indst]:=cat(w,op(state))=
  Union(Epsilon,seq(Prod(letter,trans[letter]),letter=alpha)
  ) od;
  {seq(eq[i],i=1..nbst),Mark=Epsilon,seq(letter=Atom,letter=
  alpha)} end;
```

gengram3 := **proc**(

patterns::list(list({identical(a), identical(b), identical(c), identical(d)})))

local α, *states*, *state*, *eq*, *n*, *trans*, *letter*, *nbst*, *st*, *indst*, *i*;

α := [a, b, c, d];

```
nbst := 1;
states[1] := [ ];
for indst while indst ≤ nbst do
    state := states[indst];
    n := nops(state);
    for letter in α do
        if member([op(state), letter], patterns) then
            trans[letter] := Prod(Mark, w)
        elif member([op(state), letter], map(
        proc(x, n) if n < nops(x) then [op(1 .. n + 1, x)] fi end, patterns,
        nops(state))) then
            trans[letter] := cat(w, op(state), letter);
            nbst := nbst + 1;
            states[nbst] := [op(state), letter]
        else
            for i from indst by -1 to 2 do
                st := states[i];
                if st = [op(n - nops(st) + 2 .. n, state), letter] then
                    trans[letter] := cat(w, op(st)); break
                fi
            od;
            if i = 1 then trans[letter] := w fi
        fi
    od;
    eq[indst] := cat(w, op(state)) =
        Union(E, seq(Prod(letter, trans[letter]), letter = α))
od;
{Mark = E, seq(letter = Atom, letter = α), seq(eq[i], i = 1 .. nbst)}
end
```

For instance, here is the grammar recognizing the words abab and abacab:

```
> G:=gengram3([[a,b,a,b],[a,b,a,c,a,b]]);
```

$G := \{w = \mathrm{Union}(E, \mathrm{Prod}(a, wa), \mathrm{Prod}(b, w), \mathrm{Prod}(c, w), \mathrm{Prod}(d, w)),$
$wa = \mathrm{Union}(E, \mathrm{Prod}(a, wa), \mathrm{Prod}(b, wab), \mathrm{Prod}(c, w), \mathrm{Prod}(d, w)),$
$wab = \mathrm{Union}(E, \mathrm{Prod}(a, waba), \mathrm{Prod}(b, w), \mathrm{Prod}(c, w), \mathrm{Prod}(d, w)), Mark = E,$
$a = Atom, b = Atom, c = Atom, d = Atom, waba =$
$\mathrm{Union}(E, \mathrm{Prod}(a, wa), \mathrm{Prod}(b, \mathrm{Prod}(Mark, w)), \mathrm{Prod}(c, wabac), \mathrm{Prod}(d, w)),$
$wabac = \mathrm{Union}(E, \mathrm{Prod}(a, wabaca), \mathrm{Prod}(b, w), \mathrm{Prod}(c, w), \mathrm{Prod}(d, w)),$
$wabaca = \mathrm{Union}(E, \mathrm{Prod}(a, wa), \mathrm{Prod}(b, \mathrm{Prod}(Mark, w)), \mathrm{Prod}(c, w), \mathrm{Prod}(d, w))\}$

Again, generating functions follow:

```
> subs(gfsolve(G,unlabelled,z,[[u,Mark]]),w(z,u));
```

$$-\frac{1+z^2+z^4}{-1+4z-z^2+4z^3-2z^4+z^4u+4z^5-z^6+z^6u}$$

The coefficient of $u^k z^n$ in the Taylor expansion of this rational function is the number of words of length n with k occurrences of the patterns abab and abacab. Here are the first few terms:

```
> map(expand,series(",z,10));
```

$$1+4z+16z^2+64z^3+(255+u)z^4+(1016+8u)z^5+(4048+48u)z^6+(16128+256u)z^7+(64258+1277u+u^2)z^8+(256020+6112u+12u^2)z^9+O(z^{10})$$

For instance the term $48\,u\,z^6$ corresponds to the 47 words of length 6 containing abab (ababab is counted only once), plus the word abacab itself. Of course, it would not be difficult to modify the grammar so as to take into accounts overlapping words differently. From this generating function, the treatment proceeds as before. Again, non-uniform and Markovian extensions could be considered.

```
>
```

Patterns with errors

It is also possible to accomodate patterns whose occurrence is exact except at one unspecified position. A direct way would be to apply the previous technique for multiple patterns after having generated all possible patterns obtained by introducing one error in the pattern under study. However, the number of patterns obtained this way may be much too large for this technique to be practical. A better way is to produce directly the automaton corresponding to occurrences of the pattern with at most one error, and this turns out not to be too difficult. The following procedure generates all words of length n, exact occurrences of the pattern being tagged with a mark as before (Mark0err), while occurrences with one mismatch are tagged by a Mark1err mark.

```
> gengram4:=proc
(pattern::list({identical(a),identical(b),identical(c),identical(d)})) local alpha, state, eq, n, letter, st, i, eq2, staterr, j; alpha:=[a,b,c,d]; for i to nops(pattern) do staterr[i]:={}; for letter in alpha do for j from 0 to i do if j=i or [op(1 .. i - j, pattern)] = [op(j + 1 .. i - 1, pattern), letter] then if j=0 then state[letter] := cat(w, op(1 .. i , pattern)) else staterr[i]:=staterr[i] union {[[op(1..i,pattern)], [op(1..i-j,pattern)]]}; state[letter]:=cat(w,op(1..i,pattern),`|`, op(1..i-j,pattern)) fi; break fi od; od; eq[i] := cat(w, op(1 .. i - 1, pattern)) = Union(Epsilon, seq(Prod(letter, state[letter]), letter = alpha)) od; for i to nops(pattern)-1 do for st in staterr[i] do n:=nops(st[2]); for letter in alpha do for j from 0 to n+1 do if j=n+1 or [op(1..n-j+1,pattern)]= [op(j+1..n,st[2]),letter] then if pattern[i+1]=letter then state[letter]:=cat(w,op(1..i+1,pattern),`|`, op(1..n-j+1,pattern)); staterr[i+1]:=staterr[i+1] union {[[op(1..i+1,pattern)],[op(1..n-j+1,pattern)]]} else state[letter]:=cat(w,op(1..n-j+1,pattern)) fi; break fi od od; eq2[st] := cat(w,op(st[1]),`|`,op(st[2])) = Union(Epsilon, seq(Prod(letter, state[letter]), letter = alpha)) od od; {seq(eq[i],i=1..nops(pattern)), seq(seq(eq2[st],st=staterr[i]),i=1..nops(pattern)-1),
```

```
cat(w,op(pattern))=Prod(Mark0err,w),
seq(cat(w,op(st[1]),`|`,op(st[2]))=Prod(Mark1err,w),
st=staterr[nops(pattern)]),
Mark0err=Epsilon,Mark1err=Epsilon,seq(letter=Atom,letter=a
lpha)} end;
```

```
gengram4 := proc(
pattern::list({identical(a), identical(b), identical(c), identical(d)}))
local α, state, eq, n, letter, st, i, eq2, staterr, j;
    α := [a, b, c, d];
    for i to nops(pattern) do
        staterr[i] := { };
        for letter in α do for j from 0 to i do
                if j = i or
                [op(1 .. i - j, pattern)] = [op(j + 1 .. i - 1, pattern), letter] then
                    if j = 0 then state[letter] := cat(w, op(1 .. i, pattern))
                    else
                        staterr[i] := staterr[i] union
                            {[[op(1 .. i, pattern)], [op(1 .. i - j, pattern)]]};
                        state[letter] :=
                            cat(w, op(1 .. i, pattern), `|`, op(1 .. i - j, pattern))
                    fi;
                    break
                fi
            od
        od;
        eq[i] := cat(w, op(1 .. i - 1, pattern)) =
            Union(E, seq(Prod(letter, state[letter]), letter = α))
    od;
    for i to nops(pattern) - 1 do for st in staterr[i] do
            n := nops(st[2]);
            for letter in α do for j from 0 to n + 1 do
                    if j = n + 1 or
                    [op(1 .. n - j + 1, pattern)] = [op(j + 1 .. n, st[2]), letter]
                    then
                        if pattern[i + 1] = letter then
                            state[letter] := cat(w, op(1 .. i + 1, pattern), `|`,
                                op(1 .. n - j + 1, pattern));
                            staterr[i + 1] := staterr[i + 1] union {[
                                [op(1 .. i + 1, pattern)],
                                [op(1 .. n - j + 1, pattern)]]}
                        else state[letter] := cat(w, op(1 .. n - j + 1, pattern))
                        fi;
```

```
                    break
                fi
            od
        od;
        eq2[st] := cat(w, op(st[1]), `|`, op(st[2])) =
            Union(E, seq(Prod(letter, state[letter]), letter = α))
    od
od;
{Mark0err = E, seq(letter = Atom, letter = α), Mark1err = E,
    cat(w, op(pattern)) = Prod(Mark0err, w), seq(
    cat(w, op(st[1]), `|`, op(st[2])) = Prod(Mark1err, w),
    st = staterr[nops(pattern)]),
    seq(seq(eq2[st], st = staterr[i]), i = 1 .. nops(pattern) - 1),
    seq(eq[i], i = 1 .. nops(pattern))}
end
```

Here is a simple example. The generated automaton is almost optimal (in general it has O(1) too many states):

```
> gengram4([a,b,b,c]);
```

{ $a = Atom$, $b = Atom$, $c = Atom$, $d = Atom$, $Mark0err = E$, $Mark1err = E$,

w = Union(E, Prod(a, wa), Prod(b, $wa|$), Prod(c, $wa|$), Prod(d, $wa|$)),

wa = Union(E, Prod(a, $wab|a$), Prod(b, wab), Prod(c, $wab|$), Prod(d, $wab|$)),

wab = Union(E, Prod(a, $wabb|a$), Prod(b, $wabb$), Prod(c, $wabb|$), Prod(d, $wabb|$)),

$wabb$ =

Union(E, Prod(a, $wabbc|a$), Prod(b, $wabbc|$), Prod(c, $wabbc$), Prod(d, $wabbc|$)),

$wab|a$ = Union(E, Prod(a, wa), Prod(b, $wabb|ab$), Prod(c, w), Prod(d, w)),

$wab|$ = Union(E, Prod(a, wa), Prod(b, $wabb|$), Prod(c, w), Prod(d, w)),

$wa|$ = Union(E, Prod(a, wa), Prod(b, $wab|$), Prod(c, w), Prod(d, w)),

$wabb|$ = Union(E, Prod(a, wa), Prod(b, w), Prod(c, $wabbc|$), Prod(d, w)),

$wabb|a$ = Union(E, Prod(a, wa), Prod(b, wab), Prod(c, $wabbc|$), Prod(d, w)),

$wabbc|$ = Prod($Mark1err$, w), $wabbc|a$ = Prod($Mark1err$, w),

$wabbc$ = Prod($Mark0err$, w),

$wabb|ab$ = Union(E, Prod(a, wa), Prod(b, $wabb$), Prod(c, $wabbc|$), Prod(d, w)) }

There are two kinds of states: states corresponding to words without error, which as before are represented by w.prefix, and states with one error which contain a '|', possibly followed by the last few letters which were read, when these form a prefix of the pattern. For instance, wabb|ab is the state reached by words which contain the string abb with one error, their last two letters being ab.
From there, we derive trivariate generating functions:

```
> subs(gfsolve(",unlabelled,z,[[u,Mark0err],[v,Mark1err]]),w
  (z,u,v));
```

$$-(-1-4z-6z^2+3z^8-2z^6+3z^9-6z^3-4z^4-3z^5) \Big/ (-10z^2-12z^4v-z^4u -4z^5u-30z^5v-6z^6u-21z^6v+3z^9u-5z^7u+9z^9v-6z^7v+9z^{10}v+3z^{10}u -18z^3+17z^6-7z^4-3z^8-3z^9+3z^7+21z^5+1)$$

The coefficient of $u^k\, v^l\, z^n$ in the Taylor expansion of this rational function at $z = 0$ is the number or words of length n where the pattern abbc occurs k times without error and l times with exactly one error. Again, the Markovian model could also be treated, and extensions to multiple errors are likely to be possible as well.

```
>
```

Conclusion

Various probabilistic parameters related to the occurrence of specific patterns in random words can be computed very easily using combstruct and gfun. Here, combstruct is used to model the combinatorics of the problem and gfun is very helpful to compute expansions to very large orders, thanks to the rational type of the corresponding generating functions. The model itself can be modified in various directions, to take into account different probability models or different ways of counting occurrences of the pattern when they overlap, or several patterns simultaneously.