sage-notebook

June 6, 2018

1 Computing all needed oracly queries for N = 100

1.1 Evaluations directly extracted from the notebook:

```
In [1]: oracle_queries = {
            'x': 0.36544770518929029192009234098115231104 00e-1,
            'y': 1.0,

            'R_b(x*G_1_dx(x,y),D(x*G_1_dx(x,y),y))': 'todo',
            'R_w(x*G_1_dx(x,y),D(x*G_1_dx(x,y),y))': 'todo',
            'K(x*G_1_dx(x,y),D(x*G_1_dx(x,y),y))': 'todo',
            'K_dx(x*G_1_dx(x,y),D(x*G_1_dx(x,y),y))': 'todo',

            'J_a(x*G_1_dx(x,y),D(x*G_1_dx(x,y),y))': 'todo',
            'J_a_dx(x*G_1_dx(x,y),D(x*G_1_dx(x,y),y))': 'todo',
            'G_3_arrow_dy(x*G_1_dx(x,y),D_dx(x*G_1_dx(x,y),y))': 'todo',

            'D(x*G_1_dx(x,y),y)': 1.0934784864747254921118423994176960106 08,
            'D_dx(x*G_1_dx(x,y),y)': 3.4510220643480115786480381124523971285 12,
            'P(x*G_1_dx(x,y),y)': 0.47798636953986932275273557847827380997 50e-1,
            'P_dx(x*G_1_dx(x,y),y)': 1.8025589506146115422873445288228306554 67,
            'S(x*G_1_dx(x,y),y)': 0.43615154275245713984097428006839582739 97e-1,
            'S_dx(x*G_1_dx(x,y),y)': 1.3720218102112701467662895987957102948 44,
            'H(x*G_1_dx(x,y),y)': 0.20646952454928458524714135630290468708 59e-2,
            'H_dx(x*G_1_dx(x,y),y)': .27644130352212988959440398483385617820 29,

            'G_2_dx(x*G_1_dx(x,y),y)': 0.38842683760013258631145523348425252874 0e-1,
            'G_2_dx_dx(x*G_1_dx(x,y),y)': 1.0509944030396399798033431968899880704 16,

            'G_1(x,y)': 0.37248430505369045620266187796331134907 24e-1,
            'G_1_dx(x,y)': 1.0396069237328737127831217055609221660 06,
            'x*G_1_dx(x,y)': 0.37992196457707622946696582954948631589 74e-1,
            'G_1_dx_dx(x,y)': 1.1831386535487874823112196643179832575 6,

        }
```

Moreover we have this values K, K_dx, K_dy (see below, this is not the same as our K!)

In [2]:
```
Fusy_K = 0.20646952454928458524714135630290468708859e-2
Fusy_K_dx = .21339174684114901055329420829854173799907
Fusy_K_dy = 0.18269821376205136213339189728961087053720e-1
```

1.2 Computation of evaluations that cannot be extracted from the maple notebook directly

Define variables and equations for x and y. Our only *todos* are generating functions that have to be evaluated at this weird values for x and y (this is due to the u-/l-substitutions we have in our decomposition):

In [3]:
```
var('x,y')
eqx = x==oracle_queries['x*G_1_dx(x,y)']
eqy = y==oracle_queries['D(x*G_1_dx(x,y),y)']
```

In [4]:
```
def output(var, sols):
    print str(var) + ':'
    for sol in sols:
        if sol[var].imag() == 0:
            print(sol[var])
```

1.2.1 R_b and R_w

Recall the grammar:
$R_w := (Z_U + R_b)^2$
$R_b := (Z_U + R_w)^2 \star Z_L$
From this the equations below follow directly.

In [5]:
```
var('R_w R_b')
eq1 = R_w==(y + R_b)^2
eq2 = R_b==(y + R_w)^2 * x

eqns = [eq1,eq2,eqx,eqy]
output(R_w, solve(eqns,R_b,R_w,x,y,solution_dict=True))
output(R_b, solve(eqns,R_b,R_w,x,y,solution_dict=True))
```

```
R_w:
2.577655310621243
3.072738024837374
R_b:
0.5120292887029289
0.659444182760245
```

The solution we are intereseted is the one with smallest positive real numbers.

In [6]:
```
oracle_queries['R_w(x*G_1_dx(x,y),D(x*G_1_dx(x,y),y))'] = 2.577655310621243
oracle_queries['R_b(x*G_1_dx(x,y),D(x*G_1_dx(x,y),y))'] = 0.5120292887029289
```

grafik.png

1.2.2 *K*

We do $\underline{K}$ (K_dy) first as we will need it here. We have this kind of long grammar in the paper:

In [7]:
```
var('x y R_w R_b R_w_hat R_b_hat R_w_as R_b_as K_dy')
eq1 = R_w==(y + R_b)^2
eq2 = R_b==(y + R_w)^2 * x
eq3 = R_w_hat==y+2*y*R_b+R_b^2
eq4 = R_b_hat==2*R_w_hat*x*y^2+x*R_w_hat^2
eq5 = R_w_as==2*y*R_b_hat + R_b^2
eq6 = R_b_as==2*x*y*R_w+x*R_w^2
eq7 = K_dy==R_b_as+R_w_as

eqns = [eq1,eq2,eq3,eq4,eq5,eq6,eq7,eqx,eqy]
output(K_dy, solve(eqns,R_b,R_w,R_w_hat,R_b_hat,R_w_as,R_b_as,K_dy,x,y,solution_dict=Tru
```

```
K_dy:
2.372270742358078
1.729773462783172
```

We make a little experiment here. It should be possible to obtain the value of $\underline{K}$ = K_dy by subtracting the terms that corresponds to the non-assymetric trees from the generating function ignoring the symmetrie.

In [8]:
```
# try to get K_dy with subtraction
var('x y R_w R_b K_dy')
eq1 = R_w==(y + R_b)^2
eq2 = R_b==(y + R_w)^2 * x
# explanation of this term: Figure 7 (a): the first one is impossible with this grammar
# The smallest u-size this grammar can produce is 2.
# The others are possible:
# One leaf does not count, so x*y^2 instead of x*y^3 (for the second one). The 4th has t
# different appearances when rooted at a leaf, so it's *2.
eq3 = K_dy==R_w+R_b - (x*y^2 + y^2 + 2*x*y^5)

eqns = [eq1,eq2,eq3,eqx,eqy]
output(K_dy, solve(eqns,x,y,R_w,R_b,K_dy,solution_dict=True))
```

```
K_dy:
2.372270742358078
1.729773462783172
```

In [9]:
```
K_dy = 1.729773462783172
```

It works, nice.

We have to dome some tricky stuff to get $K(x,y)$.

Lets consider the class $\widetilde{\mathcal{K}}$ of bicolored binary trees rooted at an *edge* such that the underlying unrooted tree is assymetric. As always, we let the u-size of such a tree be the number of leaves and the l-size the number of (inner) black nodes (we don't discard anything.)

For every unrooted tree $\gamma \in \mathcal{K}$ we have e distinct objects in $\widetilde{\mathcal{K}}$ where e is the number of edges. (Proof?) This works because the tree is assymetric.

It is easy to show that it holds that $e = ||\gamma|| - 3$, i.e. there are exactly 3 more leaves than edges in any of the trees in $\mathcal{K}$. (Notice that there is no problem with this equality because the trees in $\mathcal{K}$ have at least 4 leaves, any tree with 3 or less leaves is assymetric.)

Summarizing this, we have the following term for the generating function of $\widetilde{\mathcal{K}}$ (n and m are the numbers of l-atoms/u-atoms respectively):

$$\widetilde{K}(x,y) = \sum_{n,m} (m-3)\, |\mathcal{K}_{n,m}| \,\frac{x^n}{n!} y^m$$

By 3.4.2. (2) it is also true that

$$\underline{K}(x,y) = \sum_{n,m} m\, |\mathcal{K}_{n,m}| \,\frac{x^n}{n!} y^{m-1}$$

Here $\underline{\mathcal{K}}$ is the class of leaf-rooted bicolored binary trees (the root leaf doesn't count) such that the underlying tree is assymetric. For this we can already evaluate the generating function.

Now multiplying $\underline{K}(x,y)$ by y and subtracting the 2 generating functions we obtain:

$$y\underline{K}(x,y) - \widetilde{K}(x,y) = 3\sum_{n,m} |\mathcal{K}_{n,m}| \,\frac{x^n}{n!} y^m = 3K(x,y)$$

So

$$K(x,y) = \frac{1}{3}(y\underline{K}(x,y) - \widetilde{K}(x,y))$$

We can easily write down an "almost" grammar for $\widetilde{\mathcal{K}}$:

$\widetilde{\mathcal{K}} :\approx R_w \star R_b$

$R_w := (Z_U + R_b)^2$

$R_b := (Z_U + R_w)^2 \star Z_L$

This contains only one non-asymmetric tree, the fourth one of figure 7 (1 black node, 6 leaves). So we just have to subtract xy^6.

```
In [10]: var('R_w R_b Start K_snake')
         eq0 = Start==R_w*R_b
         eq1 = R_w==(y+R_b)^2
         eq2 = R_b==x*(y+R_w)^2
         eq3 = K_snake==Start - x*y^6

         eqns = [eq0,eq1,eq2,eq3,eqx,eqy]
         output(K_snake, solve(eqns,R_b,R_w,Start,K_snake,x,y,solution_dict=True))
```

```
K_snake:
1.254888507718696
1.961352378758208
```

```
In [11]: K_snake = 1.254888507718696
```

Now we can plug the values into $K(x,y) = \frac{1}{3}(y\underline{K}(x,y) - \widetilde{K}(x,y))$

```
In [12]: y = oracle_queries['D(x*G_1_dx(x,y),y)']
         K = 1/3 * (y*K_dy - K_snake)
         print('K:')
         print(K)
         oracle_queries['K(x*G_1_dx(x,y),D(x*G_1_dx(x,y),y))'] = K
```

```
K:
0.212193853436531
```

1.2.3 K' (K_dx)

From the equation for K from above it follows that

$K'(x,y) = \frac{1}{3}(y\underline{K}'(x,y) - \widetilde{K}'(x,y))$

So we have to do some derivation work. We do $\widetilde{K}'(x,y)$ first.

$\widetilde{K}'(x,y) = (R_w R_b)' - y^6$

$(R_w R_b)' = R_w' R_b + R_w R_b'$

$R_w' = 2(y + R_b)R_b'$

$R_b' = (y + R_w)^2 + 2x(y + R_w)R_w' = (y + R_w)(2xR_w' + (y + R_w))$

```
In [13]: var('x y R_w R_b R_w_dx R_b_dx Start_dx K_snake_dx ')
         eq0 = Start_dx==R_w_dx*R_b + R_w*R_b_dx
         eq1 = R_w==(y+R_b)^2
         eq2 = R_b==x*(y+R_w)^2
         eq3 = R_w_dx==2*(y+R_b)*R_b_dx
         eq4 = R_b_dx==(y+R_w)*(2*x*R_w_dx + y+R_w)
         eq5 = K_snake_dx==Start_dx - y^6

         eqns = [eq0,eq1,eq2,eq3,eq4,eq5,eqx,eqy]
         output(K_snake_dx, solve(eqns, x,y, R_w, R_b, R_w_dx, R_b_dx, Start_dx, K_snake_dx,solu
```

```
K_snake_dx:
543.8610354223433
-852.6412213740458
```

```
In [14]: K_snake_dx = 543.8610354223433
```

```
In [15]: var('x y R_w R_b R_w_dx R_b_dx K_dy_dx')
         eq1 = R_w==(y+R_b)^2
         eq2 = R_b==x*(y+R_w)^2
         eq3 = R_w_dx==2*(y+R_b)*R_b_dx
         eq4 = R_b_dx==(y+R_w)*(2*x*R_w_dx + y+R_w)
         eq5 = K_dy_dx==R_w_dx + R_b_dx - (y^2 +2*y^5)

         eqns = [eq1,eq2,eq3,eq4,eq5,eqx,eqy]
         output(K_dy_dx, solve(eqns,x,y, R_w, R_b, R_w_dx, R_b_dx, K_dy_dx,solution_dict=True))
```

```
K_dy_dx:
-716.3777777777777
539.8558558558559
```

```
In [16]: K_dy_dx = 539.8558558558559
```

```
In [17]: y = oracle_queries['D(x*G_1_dx(x,y),y)']
         K_dx = 1/3 * (y*K_dy_dx - K_snake_dx)
         print('K_dx:')
         print(K_dx)
         oracle_queries['K_dx(x*G_1_dx(x,y),D(x*G_1_dx(x,y),y))'] = K_dx
```

```
K_dx:
15.4865762511452
```

This value is kind of large so I don't know ...

1.2.4 J_a

Fusy maple worksheet (1.3.1): "K is the generating function of networks such that the associated graph, obtained by adding the root edge, is 3-connected. K is equal to M/(2x^2y)" Here, M ist the generating function of rooted 3-connected planar maps (see 1.1.1), so Fusy's M is our "M_3_arrow" which in turn is equal to our "I_a" due to the primal map bijection.

```
In [18]: x = oracle_queries['x*G_1_dx(x,y)']
         y = oracle_queries['D(x*G_1_dx(x,y),y)']
         oracle_queries['J_a(x*G_1_dx(x,y),D(x*G_1_dx(x,y),y))'] = Fusy_K * 2 *x^2 * y
```

1.2.5 J_a'

Since $2x^2yK = M$ it also holds that $M' = 4xyK + 2x^2yK'$ where the derivatives are with respect to x.

```
In [19]: oracle_queries['J_a_dx(x*G_1_dx(x,y),D(x*G_1_dx(x,y),y))'] = 4*x*y*Fusy_K + 2*x^2*y*Fus
```

1.2.6 $\underline{\vec{G}_3}$ (G_3_arrow_dy)

$M' = 2x^2K + 2x^2yK'$ with the derivates with respect to y.

```
In [20]: oracle_queries['G_3_arrow_dy(x*G_1_dx(x,y),D_dx(x*G_1_dx(x,y),y))'] = 0.5 * 2*x*Fusy_K
```

1.3 Print all oracle queries so I can copy paste them easily :D

```
In [21]: oracle_queries
```

```
Out[21]: {'D(x*G_1_dx(x,y),y)': 1.09347848647472549211184239941769601061,
          'D_dx(x*G_1_dx(x,y),y)': 3.45102206434801157864803811245239712851,
          'G_1(x,y)': 0.0372484305053690456202661877963311349072,
          'G_1_dx(x,y)': 1.03960692373287371278312170556092216601,
          'G_1_dx_dx(x,y)': 1.18313865354878748231121966431798325\76,
          'G_2_dx(x*G_1_dx(x,y),y)': 0.0388426837600132586311455233484252528 74,
          'G_2_dx_dx(x*G_1_dx(x,y),y)': 1.05099440303963997980334319688998807042,
          'G_3_arrow_dy(x*G_1_dx(x,y),D_dx(x*G_1_dx(x,y),y))': 0.00013611408589658 2,
          'H(x*G_1_dx(x,y),y)': 0.00206469524549284585247141356302904687086,
          'H_dx(x*G_1_dx(x,y),y)': 0.276441303522129889594403984833856178203,
          'J_a(x*G_1_dx(x,y),D(x*G_1_dx(x,y),y))': 6.51755944546283654118400704909007336485e-6,
          'J_a_dx(x*G_1_dx(x,y),D(x*G_1_dx(x,y),y))': 0.0010167070112147091657873345049326604810
          'K(x*G_1_dx(x,y),D(x*G_1_dx(x,y),y))': 0.212193853436531,
          'K_dx(x*G_1_dx(x,y),D(x*G_1_dx(x,y),y))': 15.4865762511452,
          'P(x*G_1_dx(x,y),y)': 0.0477986369539869322752735578478273809975,
          'P_dx(x*G_1_dx(x,y),y)': 1.80255895061461154228734452882283065547,
          'R_b(x*G_1_dx(x,y),D(x*G_1_dx(x,y),y))': 0.512029288702929,
          'R_w(x*G_1_dx(x,y),D(x*G_1_dx(x,y),y))': 2.57765531062124,
          'S(x*G_1_dx(x,y),y)': 0.0436151542752457139840974280068395827400,
          'S_dx(x*G_1_dx(x,y),y)': 1.37202181021127014676628959879571029484,
          'x': 0.0365447705189290291920092340981152311040,
          'x*G_1_dx(x,y)': 0.0379921964577076229466965829549486315897,
          'y': 1.00000000000000}
```