```
> restart: with(plots): with(gfun):
```

[18] Kilian Raschel. “Counting walks in a quadrant: a unified approach via boundary value problems”. In: Journal of the EMS 14 (2012), pp. 749–777.
[19] Kilian Raschel and Amélie Trotignon. “On walks avoiding a quadrant”. In: Electronic Journal of Combinatorics 26.P3.31 (2019), pp. 1–34.
[20] Irina Kurkova and Kilian Raschel. “Explicit expression for the generating function counting Gessel's walks”. In: Advances in Applied Mathematics 47.3 (2011), pp.414–433.

# 4. A small and a Large Compartment

## 4.1. F(x,0,t) and F(0,y,t) are interpreted as [x^{<}][y^{0}]F(x,y,t) and as [x^{0}][y^{<}]F(x,y,t) resp.

### *Generating Functions*

We define the generating functions F_{1}=F1, F_{2}=F2, F_{2}^{D}=F2D and F_{2}^{L}=F2L.

```
> count1:= proc(i,j,n)  option remember;
    if n=0 then
      if i=-1 and j=-1 then 1 else 0 fi
    else
      if i=-1 and j=-1 then count1(i, j-1, n-1) + count1(i-1,j,
  n-1);
      elif i=-1 and j<-1  then count1(i, j+1, n-1) + count1(i,
  j-1, n-1) + count1(i-1,j, n-1);
      elif j=-1 and i<-1  then count1(i+1, j, n-1) + count1(i-1,
  j, n-1) + count1(i,j-1, n-1);
      else count1(i-1, j, n-1)+ count1(i+1, j, n-1) + count1(i,
  j+1, n-1) + count1(i, j-1, n-1);
      fi
    fi
  end proc:
> F:=series(add(add(add(count1(i-1, j-1, k)*t^k*x^(i-1)*y^(j-1),
  i = -k .. k),j= -k..k), k = 0 .. 10), t,10):

  F1:=series(add(add(add(count1(i-1, j-1, k)*t^k*x^(i-1)*y^
  (j-1), i = -k .. 0),j= -k..0), k = 0 .. 10), t,10):

  F2:=series(F-F1,t):

  F2D:=series(add(add(count1(i-1, i-1, k)*t^k*x^(i-1)*y^(i-1), i
  = 1 .. k), k = 0 .. 10), t,10):

  F2L:=series(add(add(add(count1(i-1, j-1, k)*t^k*x^(i-1)*y^
  (j-1), j = -k .. i-1),i= 1..k), k = 0 .. 10), t,10):
```

### *Functional Equations*

System of functional equations for F2D and F2L

```
> expand(series(F2D-(2*t*(1/x+y)*series(add(add(count1(i-1, i-2,
  k)*t^k*x^(i-1)*y^(i-2), i = 1 .. k), k = 0 .. 10), t,10)-2*
  t/x*series(add(count1(0, -1, k)*t^k*x^(0)*y^(-1), k = 0 ..
  10),t,10)),t,10)):
> expand(series(F2L-(t*series(add(add(count1(-1, j-1, k)*t^k*y^
  (j-1),j= -k..0), k = 0 .. 10), t,10)+t*(x+1/y)*F2D+t*(x+1/x+
  y+1/y)*F2L-t*(1/x+y)*series(add(add(count1(i-1, i-2, k)*t^k*x^
```

```
   (i-1)*y^(i-2), i = 1 .. k), k = 0 .. 10), t,10)+t/x/y*series
   (add(count1(0, -1, k)*t^k*x^(0), k = 0 .. 10),t,10)-t/x*series
   (add(add(count1(0, j-1, k)*t^k*x^(0)*y^(j-1),j= -k..0), k = 0
   .. 10), t,10)),t,10)):
```

Equation for F2L

```
> expand(series(F2L*x*y*(t*(x+1/x+y+1/y)-1)-(t*y*series(add(add
  (count1(0, j-1, k)*t^k*x^(0)*y^(j-1),j= -k..0), k = 0 .. 10),
  t,10)-(t*(x^2*y+x)-x*y/2)*F2D-t*x*y*series(add(add(count1(-1,
  j-1, k)*t^k*y^(j-1),j= -k..0), k = 0 .. 10), t,10)),t,10)):
```

We multiply the last equation by -xy, and after the change of variable \varphi(x,y)=(xy,x^{-1}) [as in [19], Eq. 14] we obtain the following equation that we label by (FctEq)

Left hand side : xyF_{2, \varphi}^{L}( t (x + xy + x^{-1} + x^{-1}y^{-1}) - 1)

```
> expand(series(x*y*subs([x=x*y, y=1/x], convert(F2L,polynom))*
  subs([x=x*y, y=1/x],t*(x+1/x+y+1/y)-1),t,6)):
```

Right hand side : t [y^{0}]F_{2, \varphi}^{L} - x ( t (xy^2 + xy) - y/2) F_{2,\varphi}^{D} - t x^{2} y [1/y] F_{1,\varphi}

```
> expand(series(t*coeff(subs([x=x*y, y=1/x],convert(F2L,
  polynom)),y,0)-x*(t*(x*y^2+x*y)-y/2)*subs([x=x*y, y=1/x],
  convert(F2D, polynom))-t*x^2*y*coeff(subs([x=x*y, y=1/x],
  convert(F1, polynom)),y,-1),t,6)):
```

The polynomial xy(t*(x+xy+x^{-1}+x^{-1}y^{-1})-1) in the left hand side is called the kernel.

## *Kernel, Roots and Curve*

To ensure series convergence, we assume 0 < t < 1/4. The kernel K can be seen as a polynomial in y of degree 2 in x and and conversely.

```
> t0:=1/8:
> K:=x*y*(t*(x+x*y+x^(-1)+x^(-1)*y^(-1))-1):
> aX:=coeff(K,y,2): bX:=coeff(K,y,1): cX:=coeff(K,y,0):
  aY:=coeff(K,x,2): bY:=coeff(K,x,1): cY:=coeff(K,x,0):
  dX:=bX^2-4*aX*cX:
  dY:=expand(bY^2-4*aY*cY):
```

Branch points.
The discriminant dX has four positive roots, also called branch points such that 0 < x1 < x2 < 1 < x3 < x4. It is negative on the intervals (x1,x2) and (x3,x4).
The discriminant dY has fouthree positive roots, also called branch points such that 0 < y1 < y2 < 1 < y3. It is negative on the intervals (x1,x2) and (x3,+\infty).

```
> x3, x2, x4, x1 := allvalues(RootOf(dX, x)):
  evalf(subs(t = t0, [x1, x2, x3, x4])):#should be increasing

  y1, y3, y2 := allvalues(RootOf(dY, y)):
  evalf(subs(t = t0, [y1, y2, y3])):#should be increasing
```

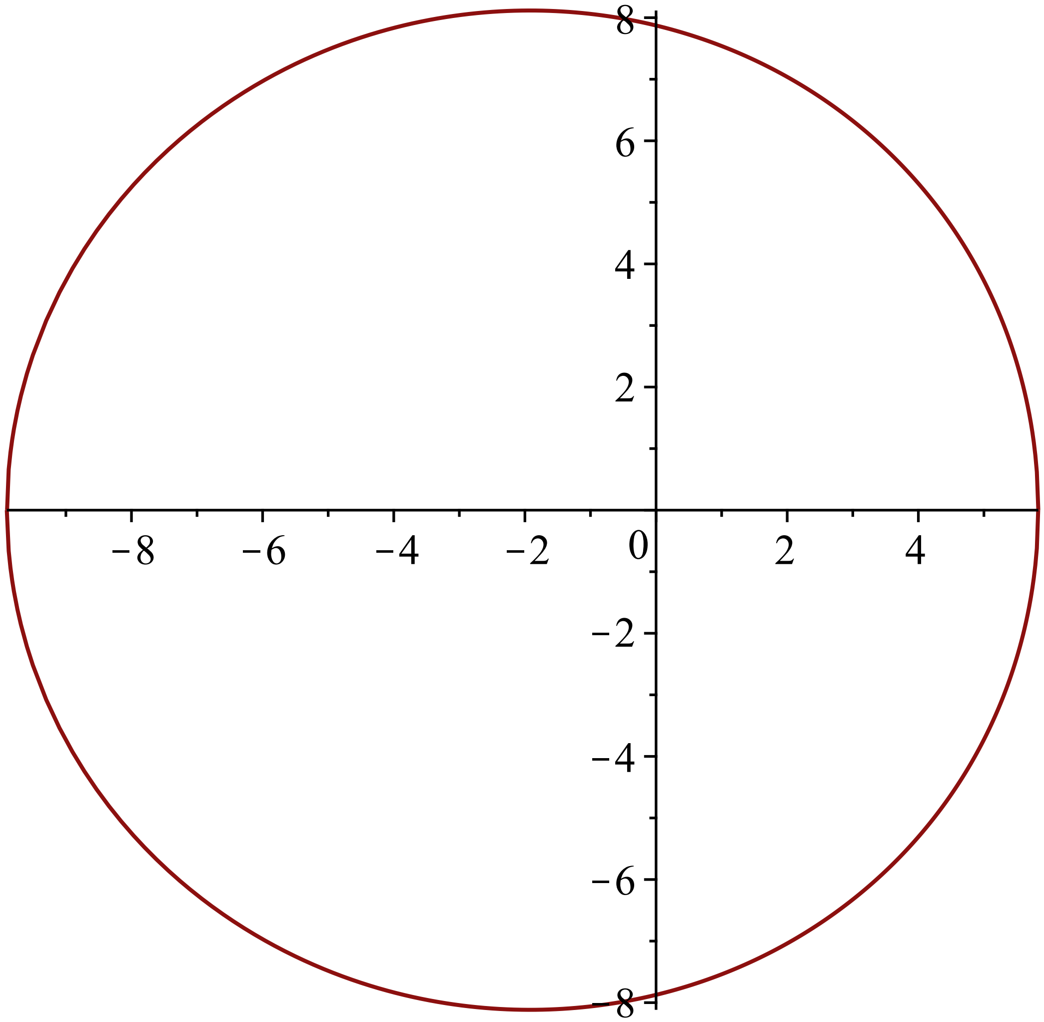
Algebraic roots of the Kernel.
Let Y be the algebraic function defined by K(x,Y(x))=0. This function has to branches, Y0=Y- and Y1=Y+, both meromorphic on the cut plane **C** \ ([x1,x2] \cup [x3,x4]).
Let X be the algebraic function defined by K(X(y),y)=0. This function has to branches, X0=X- and X1=X+, both meromorphic on the cut plane **C** \ ([y1,y2] \cup [y3,+\infty]).

```
> Y0:=(-bX-sqrt(dX))/(2*aX):
  Y1:=(-bX+sqrt(dX))/(2*aX):
  Y0s:=convert(expand(series(Y0,t)),polynom) assuming(x>0,t>0):
  Y1s:=convert(expand(series(Y1,t)),polynom) assuming(x>0,t>0):
  X0:=(-bY-sqrt(dY))/(2*aY):
  X1:=(-bY+sqrt(dY))/(2*aY):
```

Curve.

We define the curve L by L:=Y-([x1,x2])\cup Y+([x1,x2])={y\in \C : K(x,y)=0 and x\in [x1,x2]}.
Let G_L be the open domain delimited by L and avoiding the real point at +\infty

```
> L1 := complexplot(subs(t=t0,Y0), x = Re(subs(t = t0, x1)) ..
  Re(subs(t = t0, x2))):
  L2 := complexplot(subs(t=t0,Y1), x = Re(subs(t = t0, x1)) ..
  Re(subs(t = t0, x2))):
  LY:=display([L1, L2]);
```




## *Boundary Value Problem*

We want to transform the functional equation (FctEq) into a Boundary Value Problem on the curve L. The techniques are similar to the one in [19]. The interested reader in boundary value problem and walks problem can start with reading [18].

For x close to [x1,x2], we evaluate the functional equation (FctEq) at Y0.

Left hand side (it is zero because Y0 annihilates the Kernel)

```
> expand(series(subs(y=Y0,x*y*subs([x=x*y, y=1/x], convert(F2L,
  polynom))*subs([x=x*y, y=1/x],t*(x+1/x+y+1/y)-1)),t,6))
  assuming x>0:
```

Right hand side

```
> expand(series(subs(y=Y0,t*coeff(subs([x=x*y, y=1/x],convert
  (F2L, polynom)),y,0)-x*(t*(x*y^2+x*y)-y/2)*subs([x=x*y, y=
  1/x],convert(F2D, polynom))-t*x^2*y*coeff(subs([x=x*y, y=1/x],
```

```
convert(F1, polynom)),y,-1)),t,6)) assuming x>0:
```

We obtain two new equations by letting x go to any point of [x1; x2] with a positive (resp. negative) imaginary part. This two equations are complex conjugate in y. We do the subtraction of the two equations and obtain the boundary value problem: **(where \bar stand for the complex conjugation)**
\sqrt{dY(y)}/2 F_{2, \varphi}^{D}(y,t) - \sqrt{dY(\bar y)}/2 F_{2, \varphi}^{D}(\bar y,t) = t X-(y) (y - \bar y) ( [1/y]F_{1,\varphi}(x,y,t)) _{evaluated at x = X-(y)}.

### *Integral expression for F2D*

The complete solution of the boundary value problem is similar to the one stated in Theorem 7 of [10]. We have not been able to expand in series this complicated integral expression. Let us however make some remarks about this expression we wrote p.7 of our article.
-In the integral expression for D2L, we use an algebraic conformal gluing function w which has a pole in y2 with residue sqrt(r).
We can take the function 1/(w-w(y2) where w is stated in Theorem 7 of [20].
- The algebraic factor A is A(y)=-\sqrt{r} w'(y) / (i \pi \sqrt{(dY)'(y2)(w(y)-w(Y(x1)))(w(y)-w(Y(x2)))}).
- The algebraic factor B is B(y,z):=ztX0(z) w(z) / (\sqrt{w(z) - w(y1)} (w(z) - w(y)).
- The algebraic factor C is C(y):= X0(y).
- The part in the integrand is inside the curve minus the segment [x1,x2], hence by Cauchy's integral theorem, the contour L may be replaced by the unit circle.

```
> restart: with(plots): with(gfun):
```

## 4.2. F(x,0,t) and F(0,y,t) are interpreted as [x^{\leq}][y^{0}]F(x,y,t) and as [x^{0}][y^{\leq}]F(x,y,t) resp.

The method is the same as in Section 4.1 of this document, with however an additional term colored in blue.

### *Generating Functions*

### *Functional Equations*

System of functional equations for F2D and F2L. We have coloured the additional term in blue compared to section 4.1.

```
> expand(series(F2D-(2*t*(1/x+y)*series(add(add(count2(i-1, i-2, k)*t^k*x^(i-1)*y^(i-2), i = 1 .. k), k = 0 .. 10), t,10)-2*t/x*series(add(count2(0, -1, k)*t^k*x^(0)*y^(-1), k = 0 .. 10),t,10)),t,10)):
```

```
> expand(series(F2L-(t*series(add(add(count2(-1, j-1, k)*t^k*y^(j-1),j= -k..0), k = 0 .. 10), t,10)+t*(x+1/y)*F2D+t*(x+1/x+y+1/y)*F2L-t*(1/x+y)*series(add(add(count2(i-1, i-2, k)*t^k*x^(i-1)*y^(i-2), i = 1 .. k), k = 0 .. 10), t,10)+t/x/y*series(add(count2(0, -1, k)*t^k*x^(0), k = 0 .. 10),t,10)-t/x*series(add(add(count2(0, j-1, k)*t^k*x^(0)*y^(j-1),j= -k..0), k = 0 .. 10), t,10)-t/y*series(add(count2(0, 0, k)*t^k, k = 0 .. 10), t,10)),t,10)):
```

Equation for F2L

```
> expand(series(F2L*x*y*(t*(x+1/x+y+1/y)-1)-(t*y*series(add(add(count2(0, j-1, k)*t^k*x^(0)*y^(j-1),j= -k..0), k = 0 .. 10), t,10)-(t*(x^2*y+x)-x*y/2)*F2D-t*x*y*series(add(add(count2(-1,
```