Demo of Aligator

Loading the package

```
In[1]:= << Aligator.m
```

```
Aligator.m
Automated Loop Invariant Generation by Algebraic Techniques Over the Rationals.
Package written by Laura Kovacs — © RISC Linz and EPFL Lausanne — V 0.3 (2008-02-01)
```

Examples for Loops without Conditionals

Note: * stand for own examples

Simple Example 1 *

```
(*
 x:=0;y:=1;
 WHILE[…, x:=2*x;y:=1/2*y+1]
*)
```

■ Top level command: Aligator

```
In[2]:= Aligator[WHILE[…, x := 2 * x; y := 1 / 2 * y + 1]]
```

```
Method is complete!
```

```
Out[2]= x (-2 + y) == x[0] (-2 + y[0])
```

```
In[3]:= Aligator[WHILE[…, x := 2 * x; y := 1 / 2 * y + 1], IniVal → {x := 0; y := 1}]
```

```
Method is complete!
```

```
Out[3]= x y == 2 x
```

■ Inside Aligator

```
In[4]:= L = IfWhileTransform[WHILE[…, x := 2 * x; y := 1 / 2 * y + 1], Body[], Body[]]
```

$$\text{Out[4]= } \text{Body}\left[x := 2\,x;\ y := \frac{y}{2} + 1\right]$$

```
In[5]:= polys = InvLoopAssg[L]
```

```
Method is complete!
```

```
Out[5]= {x (-2 + y) - x[0] (-2 + y[0])}
```

Simple Example 2 *

```
(*
 a:=0;b:=0;
 WHILE[…, a:=a+2*b+1; b:=b+1]
*)
```

■ Top level command: Aligator

```
In[6]:= Aligator[WHILE[…, a := a + 2 * b + 1; b := b + 1]]
```

```
Method is complete!
```

```
Out[6]= a + b[0]^2 == b^2 + a[0]
```

```
In[7]:= Aligator[WHILE[…, a := a + 2 * b + 1; b := b + 1], IniVal → {a := 0; b := 0}]
```

```
Method is complete!
```

```
Out[7]= a == b^2
```

■ Inside Aligator

```
In[8]:= L = IfWhileTransform[WHILE[…, a := a + 2 * b + 1; b := b + 1], Body[], Body[]]
```

```
Out[8]= Body[a := a + 2 b + 1; b := b + 1]
```

```
In[9]:= polys = InvLoopAssg[L]
```

```
Method is complete!
```

```
Out[9]= {-a + b^2 + a[0] - b[0]^2}
```

Simple Example 3 *

```
(*
 a:=0;b:=0;c:=1;
 WHILE[…,a:=a+1;b:=b+c;c:=c+2]
*)
```

■ Top level command: Aligator

In[10]:=
```
Aligator[WHILE[…, a := a + 1; b := b + c; c := c + 2]]
```

```
Method is complete!
```

Out[10]=
$$2\,a + c[0] == c + 2\,a[0] \;\&\&\; 4\,b + 2\,c + c[0]^2 == c^2 + 4\,b[0] + 2\,c[0]$$

In[11]:=
```
Aligator[WHILE[…, a := a + 1; b := b + c; c := c + 2], IniVal → {a := 0; b := 0; c := 1}]
```

```
Method is complete!
```

Out[11]=
$$1 + 2\,a == c \;\&\&\; 4\,b == (-1 + c)^2$$

■ Inside Aligator

In[12]:=
```
L = IfWhileTransform[WHILE[…, a := a + 1; b := b + c; c := c + 2], Body[], Body[]]
```

Out[12]=
```
Body[a := a + 1; b := b + c; c := c + 2]
```

In[13]:=
```
polys = InvLoopAssg[L]
```

```
Method is complete!
```

Out[13]=
$$\{-4\,b - 2\,c + c^2 + 4\,b[0] + 2\,c[0] - c[0]^2,\; -2\,a + c + 2\,a[0] - c[0]\}$$

Simple Example 4 *

```
(*
 a:=0;b:=0;c:=1;s:=0;
 WHILE[…, a:=a+1; b:=b+c; c:=c+2; s:=s+2*a+1]
*)
```

■ Top level command: Aligator

In[14]:=
```
Aligator[WHILE[…, a == a + 1; b == b + c; c == c + 2; s == s + 2 * a + 1]]
```

```
aligator::InputError2 :
 Not appropiate Aligator-input.Input must be of the form WHILE[loop-condition, loop-
   body]. Given loop-body has to be a sequence of the form: s0; IF[test, s1, s2]; s3,
  where s0, s1, s3 are SEQUENCES OF ASSIGNMENTS, s2 is a SEQUNECE OF ASSIGNMENTS or
   IF-STATEMENT, and all conditions are NON-EMPTY BOOLEAN CONDITIONS (without :=)!
 Hint: check also for spellings (e.g. := instead of =, IF instead of If, ; instead of ,)!
```

Out[14]=
```
$Aborted
```

In[16]:=
```
Aligator[WHILE[…, a := a + 1; b := b + c; c := c + 2; s := s + 2 * a + 1]]
```

```
Method is complete!
```

Out[16]=
$$\begin{aligned} &2\,a + c[0] == c + 2\,a[0] \;\&\&\\ &\quad 2\,b + c\,(3 + 2\,a[0]) + c[0]^2 + 2\,s[0] == 2\,s + 2\,b[0] + (3 + c + 2\,a[0])\,c[0] \;\&\&\\ &\quad c^2 + 4\,c\,(1 + a[0]) + c[0]^2 + 4\,s[0] == 4\,s + 2\,(2 + c + 2\,a[0])\,c[0] \end{aligned}$$

```
In[17]:= Aligator[WHILE[…, a := a + 1; b := b + c; c := c + 2; s := s + 2 * a + 1],
          IniVal → {a := 0; b := 0; c := 1; s := 0}]
```

Method is complete!

```
Out[17]= 1 + 2 a == c && b + c == 1 + s && c (2 + c) == 3 + 4 s
```

■ Inside Aligator

```
In[18]:= L = IfWhileTransform[
            WHILE[…, a := a + 1; b := b + c; c := c + 2; s := s + 2 * a + 1], Body[], Body[]]
```

```
Out[18]= Body[a := a + 1; b := b + c; c := c + 2; s := s + 2 a + 1]
```

```
In[19]:= polys = InvLoopAssg[L]
```

Method is complete!

```
Out[19]= {c^2 - 4 s + c (4 + 4 a[0] - 2 c[0]) - 4 c[0] - 4 a[0] c[0] + c[0]^2 + 4 s[0],
          -2 b + 2 s + 2 b[0] + 3 c[0] + 2 a[0] c[0] - c[0]^2 + c (-3 - 2 a[0] + c[0]) - 2 s[0],
          -2 a + c + 2 a[0] - c[0]}
```

Division Example, by E. W. Dijkstra

```
In[20]:= (*
          rem:=x;quo:=0;
          WHILE[…,rem:=rem-y;quo:=quo+1]
         *)
```

■ Top level command: Aligator

```
In[21]:= Aligator[WHILE[…, rem := rem - y; quo := quo + 1]]
```

Method is complete!

```
Out[21]= rem + quo y == y quo[0] + rem[0]
```

```
In[22]:= Aligator[WHILE[…, rem := rem - y; quo := quo + 1], IniVal → {rem := x; quo := 0}]
```

Method is complete!

```
Out[22]= rem + quo y == x
```

■ Inside Aligator

```
In[23]:= L = IfWhileTransform[WHILE[…, rem := rem - y; quo := quo + 1], Body[], Body[]]
```

```
Out[23]= Body[rem := rem - y; quo := quo + 1]
```

In[24]:=
```
polys = InvLoopAssg[L]
```

Method is complete!

Out[24]=
$$\{\text{rem} + \text{quo}\, y - y\, \text{quo}[0] - \text{rem}[0]\}$$

Integer Square Root, by M. Kirchner - version 1

In[25]:=
```
(*
 k:=0; j:=1; m:=1;
 WHILE[m≤n, k:=k+1; j:=j+2; m:=m+j]
*)
```

■ Top level command: Aligator

In[26]:=
```
Aligator[WHILE[m ≤ n, k := k + 1; j := j + 2; m := m + j]]
```

Method is complete!

Out[26]=
$$j + 2\,k[0] == 2\,k + j[0] \;\&\&\; 2\,j + j^2 + 4\,m[0] == 4\,m + j[0]\,(2 + j[0])$$

In[27]:=
```
Aligator[WHILE[m ≤ n, k := k + 1; j := j + 2; m := m + j], IniVal → {k := 0; j := 1; m := 1}]
```

Method is complete!

Out[27]=
$$j == 1 + 2\,k \;\&\&\; (1 + j)^2 == 4\,m$$

■ Inside Aligator

In[28]:=
```
L = IfWhileTransform[WHILE[m ≤ n, k := k + 1; j := j + 2; m := m + j], Body[], Body[]]
```

Out[28]=
$$\text{Body}[k := k + 1;\, j := j + 2;\, m := m + j]$$

In[29]:=
```
polys = InvLoopAssg[L]
```

Method is complete!

Out[29]=
$$\{2\,j + j^2 - 4\,m - 2\,j[0] - j[0]^2 + 4\,m[0],\; -j + 2\,k + j[0] - 2\,k[0]\}$$

Integer Square Root, by D. E. Knuth - version 2

In[30]:=
```
(*
 y:=a/2;r:=0;
 WHILE[…,y:=y-r; r:=r+1]
*)
```

■ Top level command: Aligator

```
In[31]:= Aligator[WHILE[…, y := y - r; r := r + 1]]
```

$$\text{Out[31]}= r^2 + 2\,y + r[0] == r + r[0]^2 + 2\,y[0]$$

```
In[32]:= Aligator[WHILE[…, y := y - r; r := r + 1], IniVal → {y := a / 2; r := 0}]
```

```
Method is complete!
```

$$\text{Out[32]}= a + r == r^2 + 2\,y$$

■ Inside Aligator

```
In[33]:= L = IfWhileTransform[WHILE[…, y := y - r; r := r + 1], Body[], Body[]]
```

```
Out[33]= Body[y := y - r; r := r + 1]
```

```
In[34]:= polys = InvLoopAssg[L]
```

```
Method is complete!
```

$$\text{Out[34]}= \{-r + r^2 + 2\,y + r[0] - r[0]^2 - 2\,y[0]\}$$

Integer Cubic Root, by D. E. Knuth

```
In[35]:= (*
          x:=a;r:=1;s:=13/4;
          WHILE[…,x:=x-s; s:=s+6*r+3; r:=r+1]
         *)
```

■ Top level command: Aligator

```
In[36]:= Aligator[ WHILE[…, x := x - s; s := s + 6 * r + 3; r := r + 1]]
```

```
Method is complete!
```

$$\text{Out[36]}= 3\,r^2 + s[0] == s + 3\,r[0]^2 \;\&\&\; r + 2\,r^3 + 2\,x + r[0]^2\,(3 + 4\,r[0]) + 2\,r\,s[0] == 3\,r^2 + r[0] + 6\,r\,r[0]^2 + 2\,r[0]\,s[0] + 2\,x[0]$$

```
In[37]:= Aligator[ WHILE[…, x := x - s; s := s + 6 * r + 3; r := r + 1],
          IniVal → {x := a; r := 1; s := 13 / 4}]
```

```
Method is complete!
```

$$\text{Out[37]}= \frac{1}{4} + 3\,r^2 == s \;\&\&\; 1 + 4\,a + 6\,r^2 == 3\,r + 4\,r^3 + 4\,x$$

■ Inside Aligator

```
In[38]:= L = IfWhileTransform[WHILE[…, x := x - s; s := s + 6 * r + 3; r := r + 1], Body[], Body[]]

Out[38]= Body[x := x - s; s := s + 6 r + 3; r := r + 1]

In[39]:= polys = InvLoopAssg[L]

         Method is complete!

Out[39]= {3 r^2 - s - 3 r[0]^2 + s[0],
           r - 3 r^2 + 2 r^3 + 2 x - r[0] + 3 r[0]^2 - 6 r r[0]^2 + 4 r[0]^3 + 2 r s[0] - 2 r[0] s[0] - 2 x[0]}
```

Consecutive Cubes, by H. Cohen

```
(*
 n:=0; x:=0; y:=1; z:=6;
WHILE[(n≤N), x:=x+y; y:=y+z; z:=z+6; n:=n+1 ]
*)
```

■ Top level command: Aligator

```
Aligator[ WHILE[(n ≤ N), x := x + y; y := y + z; z := z + 6; n := n + 1 ]]

Method is complete!

6 n + z[0] == z + 6 n[0] &&
 2 n^3 + 2 x[0] + 2 n (2 + 6 n[0] + 3 n[0]^2 + y[0]) + n^2 z[0] + n[0] (1 + n[0]) z[0] ==
  6 n^2 (1 + n[0]) + 2 (x + n[0] (2 + 3 n[0] + n[0]^2 + y[0])) + n (1 + 2 n[0]) z[0] &&
 3 n^2 + 3 n[0] + 3 n[0]^2 + y[0] + n (-3 - 6 n[0] + z[0]) == y + n[0] z[0]

Aligator[ WHILE[(n ≤ N), x := x + y; y := y + z; z := z + 6; n := n + 1 ],
 IniVal → {n := 0; x := 0; y := 1; z := 6}]

Method is complete!

n^3 == x && 1 + 3 n + 3 n^2 == y && 6 + 6 n == z
```

■ Inside Aligator

```
L = IfWhileTransform[
  WHILE[(n ≤ N), x := x + y; y := y + z; z := z + 6; n := n + 1 ], Body[], Body[]]

Body[x := x + y; y := y + z; z := z + 6; n := n + 1]
```

```
polys = InvLoopAssg[L]
```

```
Method is complete!
```

$$\{-6\,n + z + 6\,n[0] - z[0],\ 3\,n^2 - y + 3\,n[0] + 3\,n[0]^2 + y[0] - n[0]\,z[0] + n\,(-3 - 6\,n[0] + z[0]),\ 2\,n^3 - 2\,x - 4\,n[0] - 6\,n[0]^2 - 2\,n[0]^3 + 2\,x[0] - 2\,n[0]\,y[0] + n\,(4 + 6\,n[0]^2 + 2\,y[0] - 2\,n[0]\,(-6 + z[0]) - z[0]) + n[0]\,z[0] + n[0]^2\,z[0] + n^2\,(-6 - 6\,n[0] + z[0])\}$$

Sum of powers ^5, by M. Petter

```
In[40]:= (*
          x:=0; y:=0;
         WHILE[(y≠k), x:=x+y^5; y:=y+1]
         *)
```

■ Top level command: Aligator

```
In[41]:= Aligator[ WHILE[(y ≠ k), x := x + y^5; y := y + 1]]
```

```
Method is complete!
```

$$\text{Out[41]=}\quad 12\,x + y^2 + 6\,y^5 + 5\,y[0]^4 + 2\,y[0]^6 == 5\,y^4 + 2\,y^6 + 12\,x[0] + y[0]^2 + 6\,y[0]^5$$

```
In[42]:= Aligator[ WHILE[(y ≠ k), x := x + y^5; y := y + 1], IniVal → {x := 0; y := 0}]
```

```
Method is complete!
```

$$\text{Out[42]=}\quad 12\,x + y^2 + 6\,y^5 == 5\,y^4 + 2\,y^6$$

■ Inside Aligator

```
In[43]:= L = IfWhileTransform[WHILE[(y ≠ k), x := x + y^5; y := y + 1], Body[], Body[]]
```

$$\text{Out[43]=}\quad \text{Body}[x := x + y^5;\ y := y + 1]$$

```
In[44]:= polys = InvLoopAssg[L]
```

```
Method is complete!
```

$$\text{Out[44]=}\quad \{-12\,x - y^2 + 5\,y^4 - 6\,y^5 + 2\,y^6 + 12\,x[0] + y[0]^2 - 5\,y[0]^4 + 6\,y[0]^5 - 2\,y[0]^6\}$$

Fibonacci Numbers * (based on the book of D. E. Knuth)

```
In[45]:= (*
          r:=1;q:=1;
          WHILE[…, t:=r; r:=r+q; q:=t]
         *)
```

■ Top level command: Aligator

```
In[46]:= Aligator[WHILE[…, t := r; r := r + q; q := t]]
```

Method is complete!

$$\text{Out[46]}= q^4 + 2\,q^3\,r + r^4 == q^2\,r^2 + 2\,q\,r^3 + (q[0]^2 + q[0]\,r[0] - r[0]^2)^2$$

```
In[47]:= Aligator[ WHILE[…, t := r; r := r + q; q := t], IniVal → {r := 1; q := 1}]
```

Method is complete!

$$\text{Out[47]}= q^4 + 2\,q^3\,r + r^4 == 1 + q^2\,r^2 + 2\,q\,r^3$$

■ Inside Aligator

```
In[48]:= L = IfWhileTransform[ WHILE[…, t := r; r := r + q; q := t], Body[], Body[]]
```

$$\text{Out[48]}= \text{Body}[t := r;\ r := r + q;\ q := t]$$

```
In[49]:= polys = InvLoopAssg[L]
```

Method is complete!

$$\text{Out[49]}= \left\{-q^4 - 2\,q^3\,r + q^2\,r^2 + 2\,q\,r^3 - r^4 + (q[0]^2 + q[0]\,r[0] - r[0]^2)^2\right\}$$

Fibonacci - Type Example * (based on the book of R. Stansley)

```
In[50]:= (*
          r:=2;q:=1;x:=2;
          WHILE[…, t:=r; r:=2 r- 8*q; q:=t;x:=8*x]
         *)
```

■ Top level command: Aligator

```
In[51]:= Aligator[ WHILE[…, t := r; r := 2 r - 8 * q; q := t; x := 8 x]]
```

Method is complete!

$$\text{Out[51]}= x\,(8\,q[0]^2 - 2\,q[0]\,r[0] + r[0]^2) == (8\,q^2 - 2\,q\,r + r^2)\,x[0]$$

```
In[52]:= Aligator[ WHILE[…, t := r; r := 2 r - 8 * q; q := t; x := 8 x],
           IniVal → {r := 2; q := 1; x := 2}]
```

Method is complete!

$$\text{Out[52]}= 8\,q^2 + r^2 == 2\,q\,r + 4\,x$$

■ Inside Aligator

```
In[53]:= L = IfWhileTransform[WHILE[…, t := r; r := 2 r - 8 * q; q := t; x := 8 * x], Body[], Body[]]

Out[53]= Body[t := r; r := 2 r - 8 q; q := t; x := 8 x]

In[54]:= polys = InvLoopAssg[L]

         Method is complete!

Out[54]= {i (-x (8 q[0]^2 - 2 q[0] r[0] + r[0]^2) + (8 q^2 - 2 q r + r^2) x[0])}
```

Tribonacci Numbers * (based on the book of R. Stansley)

```
(*
 r:=1;a:=1;b:=0;
 WHILE[…, s:=t; t:=r;r:=r+a+b;a:=t;b:=s]
*)
```

■ Top level command: Aligator

```
RecSystem[s := t; t := r; r := r + a + b; a := t; b := s]

{{r[3 + $7] == r[$7] + r[1 + $7] + r[2 + $7]},
 {{r[2 + $7], r}, {r[1 + $7], a}, {r[$7], b}}, {$7}}

Aligator[ WHILE[…, s := t; t := r; r := r + a + b; a := t; b := s]]

2 a^3 + 2 a^2 b + 2 a b^2 + b^3 - 2 a b r + b^2 r - 2 a r^2 - b r^2 + r^3 - 2 a[0]^3 - 2 a[0]^2 b[0] - 2 a[0] b[0]^2 -
  b[0]^3 + 2 a[0] b[0] r[0] - b[0]^2 r[0] + 2 a[0] r[0]^2 + b[0] r[0]^2 - r[0]^3 == 0

Aligator[ WHILE[…, s := t; t := r; r := r + a + b; a := t; b := s],
 IniVals → {r := 1; a := 1; b := 0}]

-1 + 2 a^3 + 2 a^2 b + 2 a b^2 + b^3 - 2 a b r + b^2 r - 2 a r^2 - b r^2 + r^3 == 0
```

HC polyominoes * (based on the book of R. Stansley)

```
(*
 r:=2;a:=3/4;b:=5/16;g:=1;
 WHILE[…, s:=t; t:=r;r:=5*r-7*a+4*b;a:=t;b:=s;g:=4*g]
*)
```

■ Top level command: Aligator

```
RecSystem[s := t; t := r; r := 5 * r - 7 * a + 4 * b; a := t; b := s; g := 4 * g]

{{r[3 + $10] == 4 r[$10] - 7 r[1 + $10] + 5 r[2 + $10], g[3 + $10] == 4 g[2 + $10]},
 {{r[2 + $10], r}, {r[1 + $10], a}, {r[$10], b}, {g[2 + $10], g}}, {$10}}
```

```
Aligator[ WHILE[…, s := t; t := r; r := 5 * r - 7 * a + 4 * b; a := t; b := s; g := 4 * g]]
```

$$31\,g\,a[0]^3 - 69\,g\,a[0]^2\,b[0] + 56\,g\,a[0]\,b[0]^2 - 16\,g\,b[0]^3 - 31\,a^3\,g[0] + 69\,a^2\,b\,g[0] - 56\,a\,b^2\,g[0] + 16\,b^3\,g[0] + 32\,a^2\,r\,g[0] - 47\,a\,b\,r\,g[0] + 20\,b^2\,r\,g[0] - 10\,a\,r^2\,g[0] + 7\,b\,r^2\,g[0] + r^3\,g[0] - 32\,g\,a[0]^2\,r[0] + 47\,g\,a[0]\,b[0]\,r[0] - 20\,g\,b[0]^2\,r[0] + 10\,g\,a[0]\,r[0]^2 - 7\,g\,b[0]\,r[0]^2 - g\,r[0]^3 == 0$$

Examples for Loops with Conditionals

Note: * stand for own examples

Simple Example 1 *

```
(*
 WHILE[…,
      IF[…, a:=a/2;b:=b/2;c:=c+1;d:=d+1,
            a:=a/4;b:=b/4]]
*)
```

■ Top level command: Aligator

```
In[55]:= Aligator[WHILE[…,
              IF[…, a := a / 2; b := b / 2; c := c + 1; d := d + 1,
                   a := a / 4; b := b / 4]]]
```

```
Method is complete!
```

$$\text{Out[55]= } a\,b[0] == b\,a[0] \;\&\&\; c + d[0] == d + c[0]$$

```
In[56]:= Aligator[
          WHILE[…, IF[…, a := a / 2; b := b / 2; c := c + 1; d := d + 1, a := a / 4; b := b / 4]],
          IniVal → {a := 1; b := 2; c := 0; d := 0}]
```

```
Method is complete!
```

$$\text{Out[56]= } 2\,a == b \;\&\&\; c == d$$

■ Inside Aligator

```
In[57]:= L = IfWhileTransform[WHILE[…,
               IF[…, a := a / 2; b := b / 2; c := c + 1; d := d + 1,
                    a := a / 4; b := b / 4]], Body[], Body[]]
```

$$\text{Out[57]= } \left\{\text{Body}\left[a := \frac{a}{2};\, b := \frac{b}{2};\, c := c + 1;\, d := d + 1\right],\, \text{Body}\left[a := \frac{a}{4};\, b := \frac{b}{4}\right]\right\}$$

```
In[58]:= polys = InvLoopCond[L]
```

```
        Method is complete!
```

```
Out[58]= {-c + d + c[0] - d[0], b a[0] - a b[0]}
```

Simple Example 2 *

```
In[59]:= (*
          a:=1;b:=1;c:=2;
          While[…,
           IF[…, a:=a-1; b:=b+2; c:=c-1,
                 a:=a+1; b:=b-1; c:=c-2]]
         *)
```

■ Top level command: Aligator

```
In[60]:= Aligator[While[…,
           IF[…, a := a - 1; b := b + 2; c := c - 1,
                 a := a + 1; b := b - 1; c := c - 2]]]
```

```
        aligator::InputError4 :
         Not appropiate Aligator-input.Input must be of the form WHILE[loop-condition, loop-body].
         Given input must be a WHILE loop!
```

```
Out[60]= $Aborted
```

```
In[61]:= Aligator[WHILE[…,
           IF[…, a := a - 1; b := b + 2; c := c - 1,
                 a := a + 1; b := b - 1; c := c - 2]]]
```

```
        Method is complete!
```

```
Out[61]= 5 a + 3 b + c == 5 a[0] + 3 b[0] + c[0]
```

```
In[62]:= Aligator[WHILE[…,
           IF[…, a := a - 1; b := b + 2; c := c - 1,
                 a := a + 1; b := b - 1; c := c - 2]], IniVal → {a := 1; b := 1; c := 2}]
```

```
        Method is complete!
```

```
Out[62]= 5 a + 3 b + c == 10
```

■ Inside Aligator

```
In[63]:= L = IfWhileTransform[WHILE[…,
                 IF[…, a := a - 1; b := b + 2; c := c - 1,
                  a := a + 1; b := b - 1; c := c - 2]], Body[], Body[]]
```

```
Out[63]= {Body[a := a - 1; b := b + 2; c := c - 1], Body[a := a + 1; b := b - 1; c := c - 2]}
```

```
In[64]:= polys = InvLoopCond[L]
```

```
Method is complete!
```

```
Out[64]= {-5 a - 3 b - c + 5 a[0] + 3 b[0] + c[0]}
```

Simple Example 3 *

```
In[65]:= (*
          WHILE[…,
            IF[…, x:=x+1; y:=y+1,
                   x:=x-z; y:=y+z^2;z:=z-1]]
         *)
```

■ Top level command: Aligator

```
In[66]:= Aligator[WHILE[…,
            IF[…, x := x + 1; y := y + 1,
                  x := x - z; y := y + z^2; z := z - 1]]]
```

```
Method is complete!
```

```
Out[66]= 3 y + 2 z + 3 z^2 + z^3 + 3 x[0] == 3 x + 3 y[0] + z[0] (2 + 3 z[0] + z[0]^2)
```

```
In[67]:= Aligator[WHILE[…,
            IF[…, x := x + 1; y := y + 1,
                  x := x - z; y := y + z^2; z := z - 1]], IniVal → {x := 1; y := 2; z := 0}]
```

```
Method is complete!
```

```
Out[67]= 3 (1 + x) == 3 y + z (2 + 3 z + z^2)
```

■ Inside Aligator

```
In[68]:= L = IfWhileTransform[WHILE[…,
              IF[…, x := x + 1; y := y + 1,
                    x := x - z; y := y + z^2; z := z - 1]], Body[], Body[]]
```

```
Out[68]= {Body[x := x + 1; y := y + 1], Body[x := x - z; y := y + z^2; z := z - 1]}
```

```
In[69]:= polys = InvLoopCond[L]
```

```
Method is complete!
```

```
Out[69]= {-3 x + 3 y + 2 z + 3 z^2 + z^3 + 3 x[0] - 3 y[0] - 2 z[0] - 3 z[0]^2 - z[0]^3}
```

Simple Example 4 *

```
In[70]:= (*
          WHILE[x ≥ 0,
              IF[…, x:=x+z; y:=y+1; z:=z-2,
                     x:=x+y; y:=y-2]]
         *)
```

■ Top level command: Aligator

```
In[71]:= Aligator[WHILE[x ≥ 0,
             IF[…, x := x + z; y := y + 1; z := z - 2,
                    x := x + y; y := y - 2]]]
```

```
        Method is complete!
```

```
Out[71]= True
```

Note: returned set of invariant polynomial relations is empty, i.e. {}, thus the invariant is TRUE

```
In[72]:= Aligator[WHILE[x ≥ 0,
             IF[…, x := x + z; y := y + 1; z := z - 2,
                    x := x + y; y := y - 2]], IniVal → {x := 1; y := 2; z := 0}]
```

```
        Method is complete!
```

```
Out[72]= True
```

■ Inside Aligator

```
In[73]:= L = IfWhileTransform[WHILE[x ≥ 0,
              IF[…, x := x + z; y := y + 1; z := z - 2,
                     x := x + y; y := y - 2]], Body[], Body[]]
```

```
Out[73]= {Body[x := x + z; y := y + 1; z := z - 2], Body[x := x + y; y := y - 2]}
```

```
In[74]:= polys = InvLoopCond[L]
```

```
        Method is complete!
```

```
Out[74]= {}
```

Wensley's Real Division Algorithm, by. B. Wegbreit

```
In[75]:= (*
          a:=0;b:=Q/2;d:=1;y:=0;
          WHILE[d≥Tol,
               IF[P<a+b,
                    b:=b/2;d:=d/2,
                    a:=a+b;y:=y+d/2;b:=b/2;d:=d/2
                ]
          ]
         *)
```

■ Top level command: Aligator

```
In[76]:= Aligator[WHILE[d ≥ Tol,
               IF[P < a + b,
                    b := b / 2; d := d / 2,
                    a := a + b; y := y + d / 2; b := b / 2; d := d / 2
                ]]]
```

```
        Method is complete!
```

```
Out[76]= b d[0] == d b[0] && a d + 2 b y[0] == 2 b y + d a[0] && a d[0] + 2 b[0] y[0] == 2 y b[0] + a[0] d[0]
```

```
In[77]:= Aligator[WHILE[d ≥ Tol,
               IF[P < a + b,
                    b := b / 2; d := d / 2,
                    a := a + b; y := y + d / 2; b := b / 2; d := d / 2
                ]], IniVal → {a := 0; b := Q / 2; d := 1; y := 0}]
```

```
        Method is complete!
```

```
Out[77]= 2 b == d Q && a d == 2 b y && a == Q y
```

■ Inside Aligator

```
In[78]:= L = IfWhileTransform[WHILE[d ≥ Tol,
                IF[P < a + b,
                     b := b / 2; d := d / 2,
                     a := a + b; y := y + d / 2; b := b / 2; d := d / 2
                  ]
            ], Body[], Body[]]
```

$$\text{Out[78]= } \left\{\text{Body}\left[b := \frac{b}{2}; d := \frac{d}{2}\right], \text{Body}\left[a := a + b; y := y + \frac{d}{2}; b := \frac{b}{2}; d := \frac{d}{2}\right]\right\}$$

```
In[79]:= polys = InvLoopCond[L]
```

```
        Method is complete!
```

```
Out[79]= {d b[0] - b d[0], 2 y b[0] - a d[0] + a[0] d[0] - 2 b[0] y[0], -a d + d a[0] + 2 b (y - y[0])}
```

Square Root, by K. Zuse

```
(*
   r:=a-1; q:=1; p:=1/2;
   WHILE[( 2*p*r≥err),
       IF[(2*r-2*q*p≥0),
          r:=2*r-2*q-p;q:=q+p;p:=p/2,
          r:=2*r;p:=p/2]]
*)
```

■ Top level command: Aligator

```
In[80]:= Aligator[ WHILE[( 2 * p * r ≥ err),
              IF[(2 * r - 2 * q * p ≥ 0),
                   r := 2 * r - 2 * q - p; q := q + p; p := p / 2,
                   r := 2 * r; p := p / 2]]]
```

Method is complete!

Out[80]= $q^2 + 2\,p\,r == q[0]^2 + 2\,p[0]\,r[0]$

```
Aligator[ WHILE[( 2 * p * r ≥ err),
      IF[(2 * r - 2 * q * p ≥ 0),
           r := 2 * r - 2 * q - p; q := q + p; p := p / 2,
           r := 2 * r; p := p / 2]], IniVal → {r := a - 1; q := 1; p := 1 / 2}]
```

Method is complete!

Out[84]= $\{0.359\,\text{Second}, a == q^2 + 2\,p\,r\}$

■ Inside Aligator

```
In[82]:= L = IfWhileTransform[WHILE[( 2 * p * r ≥ err),
              IF[(2 * r - 2 * q * p ≥ 0),
                   r := 2 * r - 2 * q - p; q := q + p; p := p / 2,
                   r := 2 * r; p := p / 2]], Body[], Body[]]
```

Out[82]= $\left\{\text{Body}\left[r := 2\,r - 2\,q - p;\ q := q + p;\ p := \frac{p}{2}\right], \text{Body}\left[r := 2\,r;\ p := \frac{p}{2}\right]\right\}$

```
In[83]:= polys = InvLoopCond[L]
```

Method is complete!

Out[83]= $\{-q^2 - 2\,p\,r + q[0]^2 + 2\,p[0]\,r[0]\}$

LCM & GCD computation, by E. W. Dijkstra

```
(*
 x:=a;y:=b;u:=b;v:=a;
     WHILE[x≠y,
              IF[x>y,
                       x:=x-y;v:=v+u,
                       y:=y-x; u:=u+v
               ]
      ];
*)
```

■ Top level command: Aligator

```
In[85]:= Aligator[ WHILE[x ≠ y,
                      IF[x > y,
                               x := x - y; v := v + u,
                               y := y - x; u := u + v
                       ]
             ]]
```

```
Method is complete!
```

```
Out[85]= u x + v y == u[0] x[0] + v[0] y[0]
```

```
In[86]:= Aligator[ WHILE[x ≠ y,
                      IF[x > y,
                               x := x - y; v := v + u,
                               y := y - x; u := u + v
                       ]
             ], IniVal → {x := a; y := b; u := b; v := a}]
```

```
Method is complete!
```

```
Out[86]= 2 a b == u x + v y
```

■ Inside Aligator

```
In[87]:= L = IfWhileTransform[WHILE[x ≠ y,
                       IF[x > y,
                                x := x - y; v := v + u,
                                y := y - x; u := u + v
                        ]
               ], Body[], Body[]]
```

```
Out[87]= {Body[x := x - y; v := v + u], Body[y := y - x; u := u + v]}
```

```
In[88]:= polys = InvLoopCond[L]
```

```
Method is complete!
```

```
Out[88]= {u x + v y - u[0] x[0] - v[0] y[0]}
```

Binary Product of Two Natural Numbers, by D. E. Knuth

```
In[89]:= (*
          x:=a; y:=b; z:=0;
            WHILE[( y≠0),
                IF[(Mod[y,2]=1),
                        z:=z+x; y:=y-1];
                x:=2*x; y:=y/2]
         *)
```

■ Top level command: Aligator

```
In[90]:= Aligator[ WHILE[( y ≠ 0),
              IF[(Mod[y, 2] = 1),
                        z := z + x; y := y - 1];
              x := 2 * x; y := y / 2]]
```

```
        Method is complete!
```

```
Out[90]= x y + z == x[0] y[0] + z[0]
```

```
In[91]:= Aligator[WHILE[( y ≠ 0),
              IF[(Mod[y, 2] = 1),
                        z := z + x; y := y - 1];
              x := 2 * x; y := y / 2], IniVal → {x := a; y := b; z := 0}]
```

```
        Method is complete!
```

```
Out[91]= a b == x y + z
```

■ Inside Aligator

```
In[92]:= L = IfWhileTransform[WHILE[( y ≠ 0),
               IF[(Mod[y, 2] = 1), z := z + x; y := y - 1];
               x := 2 * x; y := y / 2], Body[], Body[]]
```

$$\text{Out[92]= } \left\{\text{Body}\left[z := z + x;\ y := y - 1;\ x := 2\,x;\ y := \frac{y}{2}\right],\ \text{Body}\left[x := 2\,x;\ y := \frac{y}{2}\right]\right\}$$

```
In[93]:= polys = InvLoopCond[L]
```

```
        Method is complete!
```

```
Out[93]= {-x y - z + x[0] y[0] + z[0]}
```

Computing the floor of the square root, by E. W. Dijkstra

```
In[94]:= (* 2 nd WHILE*)
         (*
          WHILE[(q≠1),
                 IF[(r≥h),
                    q:=q/4; h:=p+q; p:=p/2; p:=p+q; r:=r-h,
                    q:=q/4; h:=p+q; p:=p/2]]
         *)
```

■ Top level command: Aligator

```
In[95]:= Aligator[ WHILE[(q ≠ 1),
                IF[(r ≥ h),
                    q := q / 4; h := p + q; p := p / 2; p := p + q; r := r - h,
                    q := q / 4; h := p + q; p := p / 2]]]
```

```
Method is complete!
```

```
Out[95]= p^2 q[0] == q (p[0]^2 + q[0] (-r + r[0]))
```

■ Inside Aligator

```
In[96]:= L = IfWhileTransform[WHILE[(q ≠ 1),
                 IF[(r ≥ h),
                     q := q / 4; h := p + q; p := p / 2; p := p + q; r := r - h,
                     q := q / 4; h := p + q; p := p / 2]], Body[], Body[]]
```

$$\text{Out[96]= } \left\{\text{Body}\left[q := \frac{q}{4};\ h := p + q;\ p := \frac{p}{2};\ p := p + q;\ r := r - h\right],\ \text{Body}\left[q := \frac{q}{4};\ h := p + q;\ p := \frac{p}{2}\right]\right\}$$

```
In[97]:= polys = InvLoopCond[L]
```

```
Method is complete!
```

```
Out[97]= {-p^2 q[0] + q (p[0]^2 + q[0] (-r + r[0]))}
```

Fermat's Factoring Problem, by D. E. Knuth

```
In[98]:= (*
          u:=2*R+1; v:=1; r:=R*R-N;
          WHILE[r≠0,
               IF[r>0,
                r:=r-v; v:=v+2,
                r:=r+u; u:=u+2
                ]]
         *)
```

■ Top level command: Aligator

```
In[99]:= Aligator[WHILE[r ≠ 0,
              IF[r > 0,
               r := r - v; v := v + 2,
               r := r + u; u := u + 2
               ]]]
```

```
        Method is complete!
```

```
Out[99]= 4 r + 2 u + v^2 + u[0]^2 + 2 v[0] == u^2 + 2 v + 4 r[0] + 2 u[0] + v[0]^2
```

```
In[100]:=
        Aligator[WHILE[r ≠ 0,
              IF[r > 0,
               r := r - v; v := v + 2,
               r := r + u; u := u + 2
               ]], IniVal → {u := 2 * R + 1; v := 1; r := R * R - N}]
```

```
        Method is complete!
```

```
Out[100]=
        4 N + 4 r + 2 u + v^2 == u^2 + 2 v
```

■ Inside Aligator

```
In[101]:=
        L = IfWhileTransform[WHILE[r ≠ 0,
              IF[r > 0,
               r := r - v; v := v + 2,
               r := r + u; u := u + 2
               ]], Body[], Body[]]
```

```
Out[101]=
        {Body[r := r - v; v := v + 2], Body[r := r + u; u := u + 2]}
```

```
In[102]:=
        polys = InvLoopCond[L]
```

```
        Method is complete!
```

```
Out[102]=
        {-4 r - 2 u + u^2 + 2 v - v^2 + 4 r[0] + 2 u[0] - u[0]^2 - 2 v[0] + v[0]^2}
```

Hardware Integer Division Program, by Z. Manna

```
In[103]:=
        (*2 nd While *)
        (*
         WHILE[(p≠1),
            ds:=ds/2; p:=p/2;
            IF[(r≥ds), r:=r-ds; q:=q+p]]
        *)
```

■ Top level command: Aligator

In[104]:=

```
Aligator[WHILE[(p ≠ 1),
  ds := ds / 2; p := p / 2;
   IF[(r ≥ ds), r := r – ds; q := q + p]]]
```

```
Method is complete!
```

Out[104]=

$$ds\,p[0] == p\,ds[0] \&\& q\,ds[0] + r\,p[0] == ds[0]\,q[0] + p[0]\,r[0] \&\& ds\,q + p\,r == ds\,q[0] + p\,r[0]$$

■ Inside Aligator

In[105]:=

```
L = IfWhileTransform[WHILE[(p ≠ 1),
  ds := ds / 2; p := p / 2;
    IF[(r ≥ ds), r := r – ds; q := q + p]], Body[], Body[]]
```

Out[105]=

$$\left\{\text{Body}\left[ds := \frac{ds}{2};\ p := \frac{p}{2};\ r := r - ds;\ q := q + p\right],\ \text{Body}\left[ds := \frac{ds}{2};\ p := \frac{p}{2}\right]\right\}$$

In[106]:=

```
polys = InvLoopCond[L]
```

```
Method is complete!
```

Out[106]=

$$\{-q\,ds[0] - r\,p[0] + ds[0]\,q[0] + p[0]\,r[0],\ p\,ds[0] - ds\,p[0],\ ds\,(-q + q[0]) + p\,(-r + r[0])\}$$

Hardware Style Division, by Z. Manna (version 2)

■ 1st Loop

In[107]:=

```
(* 1 st Loop *)
(* WHILE[a≥b, b:=2*b;c:=2*c]*)
```

■ Top level command: Aligator

In[108]:=

```
Aligator[WHILE[a ≥ b, b := 2 * b; c := 2 * c]]
```

```
Method is complete!
```

Out[108]=

$$c\,b[0] == b\,c[0]$$

■ Inside Aligator

■ 2nd Loop

In[111]:=

```
(* 2 nd Loop *)
(*WHILE[…,IF[…,a:=a-b;d:=d+c];
  b:=b/2;c:=c/2]*)
```

■ Top level command: Aligator

In[112]:=

```
Aligator[WHILE[…, IF[…, a := a - b; d := d + c];
  b := b / 2; c := c / 2]]
```

```
Method is complete!
```

Out[112]=

$$b\,c[0] == c\,b[0] \;\&\&\; d\,b[0] + a\,c[0] == a[0]\,c[0] + b[0]\,d[0] \;\&\&\; a\,c + b\,d == c\,a[0] + b\,d[0]$$

■ Inside Aligator

In[113]:=

```
L = IfWhileTransform[WHILE[…,
      IF[…, a := a - b; d := d + c];
      b := b / 2; c := c / 2], Body[], Body[]]
```

Out[113]=

$$\left\{\text{Body}\left[a := a - b;\, d := d + c;\, b := \frac{b}{2};\, c := \frac{c}{2}\right],\ \text{Body}\left[b := \frac{b}{2};\, c := \frac{c}{2}\right]\right\}$$

In[114]:=

```
polys = InvLoopCond[L]
```

```
Method is complete!
```

Out[114]=

$$\{c\,b[0] - b\,c[0],\ -d\,b[0] - a\,c[0] + a[0]\,c[0] + b[0]\,d[0],\ -a\,c + c\,a[0] + b\,(-d + d[0])\}$$

Binary Division, by A. Kaldewaij

In[115]:=

```
(*2 nd While *)
(*
 WHILE[(b≠B),
       q:=2*q;b:=b/2;
       IF[(r≥b), q:=q+1;r:=r-b]
 ]
*)
```

■ Top level command: Aligator

In[116]:=

```
Aligator[WHILE[(b ≠ B),
    q := 2 * q; b := b / 2;
   IF[(r ≥ b), q := q + 1; r := r – b]
 ]]
```

```
Method is complete!
```

Out[116]=

$$b\,q + r == b[0]\,q[0] + r[0]$$

■ Inside Aligator

In[117]:=

```
L = IfWhileTransform[WHILE[(b ≠ B),
     q := 2 * q; b := b / 2;
   IF[(r ≥ b), q := q + 1; r := r – b]
  ], Body[], Body[]]
```

Out[117]=

$$\left\{\text{Body}\left[q := 2\,q;\ b := \frac{b}{2};\ q := q + 1;\ r := r - b\right],\ \text{Body}\left[q := 2\,q;\ b := \frac{b}{2}\right]\right\}$$

In[118]:=

```
polys = InvLoopCond[L]
```

```
Method is complete!
```

Out[118]=

$$\{-b\,q - r + b[0]\,q[0] + r[0]\}$$

Extended Euclid's Algorithm, by D. E. Knuth

In[119]:=

```
(*
 a:=x;b:=y;p:=1; q:=0;r:=0;s:=1;
 WHILE[a≠b,
      IF[a>b,
           a:=a-b;p:=p-q;r:=r-s,
           b:=b-a;q:=q-p;s:=s-r
       ]
 ]
*)
```

■ Top level command: Aligator

In[120]:=

```
Aligator[WHILE[a ≠ b,
      IF[a > b,
           a := a – b; p := p – q; r := r – s,
           b := b – a; q := q – p; s := s – r
       ]
 ]]
```

Method is complete!

Out[120]=

```
a q + b[0] p[0] == b p + a[0] q[0] &&
 a s + b[0] r[0] == b r + a[0] s[0] && p s + q[0] r[0] == q r + p[0] s[0] &&
 r a[0] q[0] + p b[0] r[0] + a p[0] s[0] == r b[0] p[0] + a q[0] r[0] + p a[0] s[0] &&
 s a[0] q[0] + q b[0] r[0] + b p[0] s[0] == s b[0] p[0] + b q[0] r[0] + q a[0] s[0]
```

In[125]:=

```
Aligator[WHILE[a ≠ b,
      IF[a > b,
           a := a – b; p := p – q; r := r – s,
           b := b – a; q := q – p; s := s – r
       ]],
 IniVal → {a := x; b := y; p := 1;  q := 0; r := 0; s := 1}]
```

Method is complete!

Out[125]=

```
a q + y == b p && a s == b r + x && p s == 1 + q r && a == p x + r y && b == q x + s y
```

■ Inside Aligator

In[122]:=

```
L = IfWhileTransform[WHILE[a ≠ b,
       IF[a > b,
            a := a – b; p := p – q; r := r – s,
            b := b – a; q := q – p; s := s – r
        ]
  ] , Body[], Body[]]
```

Out[122]=

```
{Body[a := a – b; p := p – q; r := r – s], Body[b := b – a; q := q – p; s := s – r]}
```

In[123]:=

```
polys = InvLoopCond[L]
```

Method is complete!

Out[123]=

```
{–b p + a q + b[0] p[0] – a[0] q[0],
 –b r + a s + b[0] r[0] – a[0] s[0], –q r + p s + q[0] r[0] – p[0] s[0],
 –r b[0] p[0] + r a[0] q[0] + p b[0] r[0] – a q[0] r[0] – p a[0] s[0] + a p[0] s[0],
 –s b[0] p[0] + s a[0] q[0] + q b[0] r[0] – b q[0] r[0] – q a[0] s[0] + b p[0] s[0]}
```

Binary Division (with 4 branches), by E. Rodriguez Carbonell

```
In[126]:=
(*
 a:=x; b:=y;p:=1;q:=0;
 WHILE[((a≠0) ∧ (b≠0)),
      IF[(Mod[a,2]=0) ∧ (Mod[b,2]=0), a:=a/2;b:=b/2;p:=4*p,
           IF[(Mod[a,2]=1) ∧ (Mod[b,2]=0), a:=a-1;q:=q+b*p,
                IF[(Mod[a,2]=0) ∧ (Mod[b,2]=1),
                     b:=b-1;q:=q+a*p,
                     q:=q+(a+b-1)*p;
          a:=a-1;b:=b-1;q:=q+(a+b-1)*p ]]]]
*)
```

■ Top level command: Aligator

```
In[127]:=
Aligator[WHILE[((a ≠ 0) ∧ (b ≠ 0)),
      IF[(Mod[a, 2] = 0) ∧ (Mod[b, 2] = 0), a := a / 2; b := b / 2; p := 4 * p,
           IF[(Mod[a, 2] = 1) ∧ (Mod[b, 2] = 0), a := a - 1; q := q + b * p,
                IF[(Mod[a, 2] = 0) ∧ (Mod[b, 2] = 1),
                     b := b - 1; q := q + a * p,
                     q := q + (a + b - 1) * p; a := a - 1; b := b - 1 ]]]]]
```

```
Method is complete!
```

```
Out[127]=
a b p + q == a[0] b[0] p[0] + q[0]
```

```
In[128]:=
Aligator[WHILE[((a ≠ 0) ∧ (b ≠ 0)),
      IF[(Mod[a, 2] = 0) ∧ (Mod[b, 2] = 0), a := a / 2; b := b / 2; p := 4 * p,
           IF[(Mod[a, 2] = 1) ∧ (Mod[b, 2] = 0), a := a - 1; q := q + b * p,
                IF[(Mod[a, 2] = 0) ∧ (Mod[b, 2] = 1),
                     b := b - 1; q := q + a * p,
                     q := q + (a + b - 1) * p; a := a - 1; b := b - 1 ]]]],
 IniVal → {a := x; b := y; p := 1; q := 0}]
```

```
Method is complete!
```

```
Out[128]=
a b p + q == x y
```

■ Inside Aligator

In[129]:=

```
L = IfWhileTransform[WHILE[((a ≠ 0) ∧ (b ≠ 0)),
      IF[(Mod[a, 2] = 0) ∧ (Mod[b, 2] = 0), a := a / 2; b := b / 2; p := 4 * p,
        IF[(Mod[a, 2] = 1) ∧ (Mod[b, 2] = 0), a := a - 1; q := q + b * p,
          IF[(Mod[a, 2] = 0) ∧ (Mod[b, 2] = 1),
            b := b - 1; q := q + a * p,
            q := q + (a + b - 1) * p; a := a - 1; b := b - 1 ]]]] , Body[], Body[]]
```

Out[129]=

$$\left\{\text{Body}\left[a := \frac{a}{2}; b := \frac{b}{2}; p := 4\,p\right], \text{Body}[a := a - 1; q := q + b\,p],\right.$$
$$\left.\text{Body}[b := b - 1; q := q + a\,p], \text{Body}[q := q + (a + b - 1)\,p; a := a - 1; b := b - 1]\right\}$$

In[130]:=

```
polys = InvLoopCond[L]
```

```
Method is complete!
```

Out[130]=

$$\{-a\,b\,p - q + a[0]\,b[0]\,p[0] + q[0]\}$$

Large Number's Factorization, by D. E. Knuth

In[132]:=

```
(*
 WHILE [((s≥d)∧(r≠0)),
   IF [(2*r-rp+q<0), t:=r; r:=2*r-rp+q+d+2; rp:=t; q:=q+4;d:=d+2,
           IF [
    ((2*r-rp+q≥0) ∧ (2*r-rp+q<d+2)), t:=r;r:=2*r-rp+q;rp:=t;d:=d+2,
               IF [((2*r-rp+q≥0)∧(2*r-rp+q≥d+2) ∧(2*r-rp+q<2*d+4)),
     t:=r; r:=2*r-rp+q-d-2; rp:=t; q:=q-4; d:=d+2,
                          t:=r; r:=2*r-rp+q-2*d-4; rp:=t; q:=q-8;d:=d+2 ]]]]
*)
```

■ Top level command: Aligator

In[137]:=

```
Aligator[WHILE [((s ≥ d) ∧ (r ≠ 0)),
   IF [(2 * r - rp + q < 0), t := r; r := 2 * r - rp + q + d + 2; rp := t; q := q + 4; d := d + 2,
           IF [
    ((2 * r - rp + q ≥ 0) ∧ (2 * r - rp + q < d + 2)), t := r; r := 2 * r - rp + q; rp := t; d := d + 2,
               IF [((2 * r - rp + q ≥ 0) ∧ (2 * r - rp + q ≥ d + 2) ∧ (2 * r - rp + q < 2 * d + 4)),
     t := r; r := 2 * r - rp + q - d - 2; rp := t; q := q - 4; d := d + 2,
                          t := r; r := 2 * r - rp + q - 2 * d - 4; rp := t; q := q - 8; d := d + 2
```

```
Method is complete!
```

Out[137]=

$$2\,d\,(q + 2\,r - 2\,rp) + d[0]^2\,q[0] + 8\,r[0] + 4\,d[0]\,rp[0] == d^2\,q + 8\,r + 2\,d[0]\,(q[0] + 2\,r[0])$$

■ Inside Aligator

In[134]:=

```
L = IfWhileTransform[WHILE[((s ≥ d) ∧ (r ≠ 0)),
    IF[(2 * r - rp + q < 0), t := r; r := 2 * r - rp + q + d + 2; rp := t; q := q + 4; d := d + 2,
            IF[
      ((2 * r - rp + q ≥ 0) ∧ (2 * r - rp + q < d + 2)), t := r; r := 2 * r - rp + q; rp := t; d := d + 2,
                IF[((2 * r - rp + q ≥ 0) ∧ (2 * r - rp + q ≥ d + 2) ∧ (2 * r - rp + q < 2 * d + 4)),
     t := r; r := 2 * r - rp + q - d - 2; rp := t; q := q - 4; d := d + 2,
                                t := r; r := 2 * r - rp + q - 2 * d - 4; rp := t; q := q - 8; d := d + 2
  Body[], Body[]]
```

Out[134]=

```
{Body[t := r; r := 2 r - rp + q + d + 2; rp := t; q := q + 4; d := d + 2],
 Body[t := r; r := 2 r - rp + q; rp := t; d := d + 2],
 Body[t := r; r := 2 r - rp + q - d - 2; rp := t; q := q - 4; d := d + 2],
 Body[t := r; r := 2 r - rp + q - 2 d - 4; rp := t; q := q - 8; d := d + 2]}
```

In[135]:=

```
polys = InvLoopCond[L]
```

```
Method is complete!
```

Out[135]=

$$\{-d^2\, q - 8\, r + 2\, d\, (q + 2\, r - 2\, rp) - 2\, d[0]\, q[0] + d[0]^2\, q[0] + 8\, r[0] - 4\, d[0]\, r[0] + 4\, d[0]\, rp[0]\}$$