Re: "Collatz 3n+1 conjecture is unprovable" paper; one last comment

Source: http://sci.tech−archive.net/Archive/sci.math/2006−09/msg05632.html

- *From*: "Proginoskes" <CCHeckman@xxxxxxxxx>

- *Date*: 27 Sep 2006 22:48:13 −0700

Craig Feinstein wrote:

> Proginoskes wrote:
>
> > mensanator@xxxxxxxxxxx wrote:
> >
> > > Craig Feinstein wrote:
> > >
> > > > mensanator@xxxxxxxxxxx wrote:
> > > >
> > > > > Craig Feinstein wrote:
> > > > >
> > > > > > mensanator@xxxxxxxxxxx wrote:
> > > > > >
> > > > > > > Craig Feinstein wrote:
> > > > > > >
> > > > > > > > Proginoskes wrote:
> > > > > > > >
> > > > > > > > > A while back (Mon, May 15 2006 8:50 am), Craig Feinstein wrote in sci.math:

if the Theorem 2 [in his paper] is wrong, then you should be able to pick a number like 7 or 32 and rigorously prove that when applied to the Collatz algorithm, the algorithm will halt at one *without talking about what the algorithm does at each iteration*. If you can

do this, then and only then have you shown that Theorem 2 is wrong. If not, then you are simply fooling yourself.

Searching through MathSciNet for a paper on Collatz and the 2−adics, I found the following paper:

Stefan Andrei, Cristian Masalagiu, "About the Collatz conjecture." Acta Informatica

35
(1998),
no.
2,
167−−179.

In
the
abstract,
it
states:
"[The
authors]
show
that
the
value
of
k
such
that
$f^{(k)}(n)
=1$
can
be
found
by
an
algorithm
faster
than
the
one
deriving
from
a
direct
application
of
the
definition.
They
argue,
but
don't
give
a
definite
proof,
that
their

algorithm is three times faster than the trivial one (the one obtained by applying the definitions)."

So not only can it be proved that f^(k)(n) = 1 without using the definition, there's actually a quicker way without the definition.

OK, I'll believe you if you can use the

results
given
in
this
paper
to
rigorously
prove
that
when
the
number
7
is
applied
to
the
Collatz
algorithm,
the
algorithm
will
halt
at
one
*without
talking
about
what
the
algorithm
does
at
each
iteration*.

I'll
use
my
own
paper
instead.
Not
that
it
matters,
once
you're
proved
wrong

it
doesn't
matter
if
other
prrofs
exist.

But
I'm
not
holding
my
breath
for
such
a
proof.

##
Feinstein:
##
##
Theorem
2:
If
k,
n

N
and
T(k)(n)
=
1,
then
in
order
to
prove
##
that
T(k)(n)
=
1,
it
is
necessary
to

specify
the
values
##
of
(n,
T(n),
...,
T(k‘)(n))
mod
2
in
the
proof.
##
##
Proof:
The
formula
for
T(k)(n)
is
determined
by
the
values
of
##
(n,
T(n),
...,
T(k‘)(n))
mod
2,
and
there
is
a
##
one−to−one
correspondence
between
all
of
the
possible
##
formulas
for
T(k)(n)
and

all
of
the
possible
values
of
##
(n,
T(n),
...,
T(k‘)(n))
mod
2
(Lagarias,
1985);
##
therefore,
in
order
to
prove
that
T(k)(n)
=
1,
it
is
##
necessary
to
specify
the
values
of
(n,
T(n),
....,
##
T(k‘)(n))
mod
2
in
the
proof,
since
in
order
to
prove
##
that

```
T(k)(n)
=
1,
it
is
necessary
to
specify
the
formula
##
for
T(k)(n)
in
the
proof.

import
gmpy

#
#
Sequence
Vector
utilities
#
def
seed(a,xyz):
"""calculate
Seed
from
Hailstone

seed(a,xyz)
a:
hailstone
xyz:
HailstoneFunctionParameters
x
*
a
−
z
g
=
−−−−−−−−−
y
returns
(0)
if
invalid
```

returns
Seed
(g)
"""
g
=
divmod(xyz[0]*a−xyz[2],xyz[1])
if
g[1]==0:
return
g[0]
else:
return
0

def
gen0(k,xyz):
"""Find
first
member
of
generation
k
of
Hailstone
Function.

gen0(k,xyz)
k:
generation
xyz:
HailstoneFunctionParameters

<Non−recursive
version>

a:
the
first
solution
to
the
Hailstone
Function
g:
seed
that
genertes
_a_
d:
difference

```
between
_a_
and
_g_
j:
index
of
_a_
where
d
==
0
(mod
y**k)
c:
a[j]
the
returns
[0,0,0,0]
if
k
invalid
returns
GenerationParameters
[a,g,d,c]
"""
ONE
=
gmpy.mpz(1)
yx
=
gmpy.mpz(xyz[1]−xyz[0])
gen
=
gmpy.mpz(0)
while
gen<k:
if
gen==0:
a
=
gmpy.divm(xyz[2],xyz[0],xyz[1])
g
=
seed(a,xyz)
d
=
a
−
g
c
```

```
=
a
gen
+=
1
ap
=
a
gp
=
g
dp
=
d
cp
=
c
else:
j
=
gmpy.divm(xyz[1]**(gen)−dp,yx,xyz[1]**(gen))/xyz[1]**(ge
a
=
j*xyz[1]**(gen)
+
cp
g
=
seed(a,xyz)
c
=
a
d
=
a
−
g
gen
+=
1
ap
=
a
gp
=
g
dp
=
d
cp
=
```

```
c
return
[a,g,d,c]

def
geni(k,i,xyz):
"""Find
ith
kth
generation
hailstone

geni(k,i,xyz)
i:
member
of
generation
k:
generation
xyz:
HailstoneFunctionParameters
a(k,i)
=
i*y**k
+
a(k,0)
k,i:
{
1,
2,
3,
...}
returns
Hailstone
(a)
"""
K
=
gmpy.mpz(k)
I
=
gmpy.mpz(i−1)
if
(k<1)
or
(i<1):
return
0
else:
a
=
```

```
gen0(k,xyz)
return
I*xyz[1]**K
+
a[0]

def
calc_xyz(sv):
"""calculate
Hailstone
Function
Parameters

calc_xyz(sv)
sv:
sequence
vector
returns
HailstoneFunctionParameters
(x,y,z)
"""
TWO
=
gmpy.mpz(2)
TWE
=
gmpy.mpz(3)
twee
=
gmpy.mpz(len(sv))
twoo
=
gmpy.mpz(sum(sv))
x
=
TWO**twoo
y
=
TWE**twee
z
=
gmpy.mpz(0)
for
i
in
xrange(len(sv)):
z
=
z
+
(TWE**i
```

```
*
TWO**sum(sv[:−(i+1)]))
return
(x,y,z)


#
#
Partition
Generator
#

def
move_col(c):
sv[c−1]
+=
1
sv[c]
−=
1

def
find_c():
i
=
−1
while
i<0:
if
sv[i]>1:
return
i
i
−=
1

def
rollover(c):
move_col(c)
sv[−1]
=
sv[c]
sv[c]
=
1


#
#
Test
if
```

```
partition
generated
is
an
actual
Collatz
Sequence
#
without
actually
running
it
through
the
Collatz
#
process
(no
intermediate
values
calculated)

def
print_sv(sv):
#
All
Sequence
Vectors
that
are
Collatz
Sequences
must
end
#
in
an
even
number
>=4
#
#
This
last
number
is
appended
to
the
vector
created
```

```
by
the
#
partition
generator,
so
actual
vector
checked
will
always
#
be
width+1.
#
sv.append(4)
xyz
=
calc_xyz(sv)
a0
=
geni(1,1,xyz)
if
a0==1:
#
If
true,
Sequence
Vector
is
an
actual
Collatz
Sequence
#
that
ends
at
1,
so
find
the
seed
that
generates
it.
g0
=
seed(a0,xyz)
print
sv,
```

```
g0
return
1
return
0

#
#
Main:
prove
the
Collatz
Sequence
7
...
1
without
ever
calculating
#
any
intermediate
values
#

width
=
4

for
depth
in
xrange(width,width+9):
#
#
Partition
generator
creates
all
partitions
of
depth
items
into
#
width
containers
such
that
each
container
```

```
has
at
least
1
item.
#
This
is
equivalent
to
all
Collatz
Sequences
starting
with
an
odd
#
number
having
width
3n+1
iterations
and
depth
n/2
iterations.
#
m
=
depth
−
1
n
=
width
−
1
print
print
'Checking
%d
Sequence
Vectors
with
Width=%d
Depth=%d'
%
(gmpy.comb(m,n),width,depth)
max_element
=
```

```
depth
−
width
+
1
sv
=
[]
for
i
in
range(width):
sv.append(1)
sv[−1]
=
max_element
found
=
print_sv(sv[:])
count
=
1
while
sv[0]<max_element:
c
=
find_c()
if
c
<
−1:
rollover(c)
found
+=
print_sv(sv[:])
else:
move_col(c)
found
+=
print_sv(sv[:])
count
+=
1
print
'Solutions
found:
%d'
%
(found)

##
```

```
Checking
1
Sequence
Vectors
with
Width=4
Depth=4
##
Solutions
found:
0
##
##
Checking
4
Sequence
Vectors
with
Width=4
Depth=5
##
Solutions
found:
0
##
##
Checking
10
Sequence
Vectors
with
Width=4
Depth=6
##
Solutions
found:
0
##
##
Checking
20
Sequence
Vectors
with
Width=4
Depth=7
##
[1,
1,
2,
3,
```

```
4]
7
##
Solutions
found:
1

##
QED

##
##
Checking
35
Sequence
Vectors
with
Width=4
Depth=8
##
[1,
1,
1,
5,
4]
15
##
Solutions
found:
1
##
##
Checking
56
Sequence
Vectors
with
Width=4
Depth=9
##
[3,
1,
2,
3,
4]
29
##
Solutions
found:
1
##
```

```
##
Checking
84
Sequence
Vectors
with
Width=4
Depth=10
##
[3,
1,
1,
5,
4]
61
##
Solutions
found:
1
##
##
Checking
120
Sequence
Vectors
with
Width=4
Depth=11
##
[5,
1,
2,
3,
4]
117
##
Solutions
found:
1
##
##
Checking
165
Sequence
Vectors
with
Width=4
Depth=12
##
[2,
5,
```

2,
3,
4]
241
##
[5,
1,
1,
5,
4]
245
##
Solutions
found:
2

You
can
breathe
now.

You have
not
presented a
rigorous
proof.

Haven't you read my paper?

Blueprint for Failure
How to Construct a
Counterexample to the
Collatz Conjecture
Full paper at S.A.T.O.
Volume 5.3. (2006)
<http://home.zonnet.nl/galien8>

It's the least you can do.
After all, I read yours.

Your paper is irrelevant to this discussion.

Yes, it is. The Hailstone Function is exactly the thing you claim can't exist: a means of determing whether a starting number reaches 1 without stepping through the intermediate values. [...]

> There is another way to do this, of course. In a post a ways back, I posted a formula for the Collatz function using only +,*,−,/ and the floor function. (The last gets rid of the definition by cases.) If you iterate this, you get a finite formula for T^(k)(n). (Feinstein calls the Collatz function T.)
>
> I posted the following the thread "A Formula for T^k(n): Was: Re: Another Reason Why Collatz is Unprovable" in sci.math:
>
> > PROPOSITION 1. For any integer n, T(n) = n/2 + (n/2 − floor(n/2)) * 2 *
> > (2n+1)/2.
> > [Proof omitted]
> >
> > PROPOSITION 2. For any integer n, and any nonnegative integer k, there
> > is a finite formula for T^k(n), using the arithmetic operations and the
> > floor function.
> > [Proof by composing the formula in Proposition 1.]
> >
> > PROPOSITION 3. There is a formula F such that F(n,k) = T^k(n), for all
> > integers n, and all nonnegative integers k.
> > [Proof: F(n,k) = sum(T(n,i)*(1 − ceiling(arctan(abs(k − i)))), i=0..infinity),
> > where T(n,k) is given by Proposition 2.]
>
> All of this is just a cheap disguise, just like Mensanator's attempt. In order to actually compute these formulas, you have to determine within the computations what the Collatz algorithm does at each iteration.

Not when evaluating T(n,8); the values T(n,1), T(n,2), ..., T(n,7) are NOT calculated when T(n,8) is evaluated. And your original claim was:

> OK, I'll believe you if you can [...]
> rigorously prove that when the number 7 is applied to the Collatz algorithm, the algorithm will halt at one *without talking about what the algorithm does at each iteration*.

The formula for T(n,8) does not make use of the value of T(n,3).

Another way to construct the formulas above provides a value for
T(n,2^k) for general k:

T(n,1) = T(n)
T(n,2^(k+1)) = T(T(n, 2^k), 2^k), i.e.,

T(n,2) = T(T(n,1),1)
T(n,4) = T(T(n,2),2)
T(n,8) = T(T(n,4),4)

where (for instance) T(n,3) is not used to determine T(n,8).

> For instance, computing (n/2 − floor(n/2)) in proposition 1
> is equivalent to determining whether n is odd or even, which is really
> what determines the formula for the Collatz function.

And that's why the formula gives the right value. However, (and this is
the point that you keep ignoring) your proof doesn't know this. Your
proof claims that T^(k)(n) has to be described in terms of 2^k cases.
My non−cases, single formula for T(n,k) proves this is wrong.

The fact that there's a difference between what we know to be true and
what the proof claims to be true means the proof is wrong. So all your
clever observations only bury the proof deeper in the ground; they do
not redeem it.

−−− Christopher Heckman

> You are never going to hide the parity vector from any proof that any
> particular number converges to 1 via the Collatz algorithm.

.