

CoCoA-5 - Bug #1406

Poor memory management?

28 Jan 2020 15:56 - John Abbott

Status:	New	Start date:	28 Jan 2020
Priority:	Normal	Due date:	
Assignee:		% Done:	0%
Category:	bug	Estimated time:	0.00 hour
Target version:	CoCoA-5.4.4	Spent time:	0.45 hour
<div>Description</div> <p>I tried something like the following:</p> <pre>H := 10000; foreach C in (0..H)><(-H..H) do // something endforeach; println "Starting loop 2"; for i := 1 to 1000000 do // something slow endfor;</pre> <p>The big cartesian product in the first loop consumed about 16Gb RAM on my machine. Strangely, that memory allocation seemed to be held even after the first loop had finished.</p> <p>Why? This is undesirable!</p>			

History

#1 - 19 Mar 2021 14:46 - John Abbott

Here is a concrete example... a bit faster, but still sufficient to exhibit the problem.

```
H := 5000;
foreach C in (0..H)><(-H..H) do
  // something
endforeach;
println "Starting loop 2";
for i := 1 to 30000 do
  if factorial(i) = 120 then println i; endif;
endfor;
```

Note that, according to top, during 1st loop memory increases a bit beyond 4Gbytes, but during the 2nd loop it drops to about 3.7Gbytes (rather than 0 Gbytes). Why?

#2 - 19 Mar 2021 14:47 - John Abbott

- Target version changed from CoCoA-5.4.0 to CoCoA-5.4.2

#3 - 19 Mar 2021 15:47 - John Abbott

This is what valgrind produced:

```
With CoCoALib and external libraries GMP, Normaliz, Frobbly, Gfan, CDD, MathSAT
indent(VersionInfo(), 2); -- for information about this version
# H := 5000;
# foreach C in (0..H)><(-H..H) do
[[foreach]] #    // something
[[foreach]] # endforeach;
==14306== Warning: set address range perms: large range [0x1a627f040, 0x1c627f040) (undefined)
==14306== Warning: set address range perms: large range [0xde27e028, 0xee27e058) (noaccess)
==14306== Warning: set address range perms: large range [0x1a627f028, 0x1c627f058) (noaccess)
# println "Starting loop 2";
Starting loop 2
# for i := 1 to 30000 do
[[For]] #    if factorial(i) = 120 then println i; endif;
[[For]] # endfor;
5
# ==14306==
==14306==  HEAP SUMMARY:
==14306==      in use at exit: 134,751 bytes in 217 blocks
==14306==    total heap usage: 150,887,968 allocs, 150,887,751 frees, 8,190,471,597 bytes allocated
==14306==
==14306==  LEAK SUMMARY:
==14306==    definitely lost: 0 bytes in 0 blocks
==14306==    indirectly lost: 0 bytes in 0 blocks
==14306==    possibly lost: 0 bytes in 0 blocks
==14306==    still reachable: 134,751 bytes in 217 blocks
==14306==         suppressed: 0 bytes in 0 blocks
==14306== Rerun with --leak-check=full to see details of leaked memory
==14306==
==14306== For lists of detected and suppressed errors, rerun with: -s
==14306== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

Does not help to isolate the problem. I wonder what the 135k of still reachable memory is?
Maybe I should try with MEMPOOL_DEBUG? But not today.

#4 - 15 Jan 2024 19:51 - John Abbott

- *Target version changed from CoCoA-5.4.2 to CoCoA-5.4.4*