CoCoALib - Bug #1458

Redesign interrupt mechanism?

10 May 2020 12:06 - John Abbott

Status:	Rejected	**Start date:**	10 May 2020
Priority:	Normal	**Due date:**	
Assignee:	John Abbott	**% Done:**	100%
Category:	Improving	**Estimated time:**	0.51 hour
Target version:	CoCoALib-0.99800	**Spent time:**	0.50 hour

Description

I have just made SmoothFactor interruptible. Testing revealed disappointing performance.

```
N := factorial(1000000);
facs := SmoothFactor(N,1000000); --> then manually interrupt... TAKES ALMOST 9s!
```

Some delay is inevitable, but 9s was long enough that I originally thought the interrupt mechanism was not working.

The situation is somewhat worse if I do the following:

```
N := factorial(1000);
facs := SmoothFactor(N,1000000);
N := factorial(1000000);
facs := SmoothFactor(N,1000000); --> manually interrupt... TOOK MORE THAN 1100s
```

I thought 9s delay was bad... oh ho!

Fix this!!

Related issues:

Related to CoCoALib - Feature #1457: Make SmoothFactor interruptible	**Closed**	**10 May 2020**
Related to CoCoALib - Design #1086: New design for interrupt mechanism	**Closed**	**30 Jun 2017**
Related to CoCoALib - Bug #971: CheckForInterrupt does not work in the expect...	**Closed**	**14 Nov 2016**
Related to CoCoALib - Feature #714: Interrupt mechanism	**Closed**	**19 May 2015**

History

#1 - 10 May 2020 12:06 - John Abbott

- Related to Feature #1457: Make SmoothFactor interruptible added

#2 - 10 May 2020 12:06 - John Abbott

- Related to Design #1086: New design for interrupt mechanism added

#3 - 10 May 2020 12:06 - John Abbott

- Related to Bug #971: CheckForInterrupt does not work in the expected way added

#4 - 10 May 2020 12:07 - John Abbott

- Related to Feature #714: Interrupt mechanism added

#5 - 10 May 2020 12:14 - John Abbott

- Status changed from New to In Progress

- % Done changed from 0 to 10

The problem lies in the design.

To avoid performing a potentially costly check too often, I copied the mechanism from (CpuTimeOut?) so that actual checks are performed only rarely, with the frequency of future checks based on the frequency of checks in the "recent past".

What happens in the second example is that the first call to SmoothFactor make checks relatively quickly (because the other operations are quick). Then the second call to SmoothFactor suddenly makes checks much more slowly (presumably about 1000 times more slowly). It is wrong to use the frequency of checks in the first call to dictate the frequency of checks in the second call.

There needs to be a mechanism similar to myPrepareForNewLoop for interrupt checks: each fn which wants to be interruptible should call this at the start (and potentially inside itself if there are several loops which might iterate at different speeds).

#6 - 10 May 2020 12:16 - John Abbott

- Description updated

#7 - 10 May 2020 12:22 - John Abbott

- Description updated

#8 - 16 Jun 2020 16:45 - John Abbott

- Status changed from In Progress to Rejected

- % Done changed from 10 to 100

What I wrote in comment 5 was perhaps once correct, but not any longer.
This bug did not exist where I thought it existed. --> **REJECTING**

#9 - 07 Mar 2022 10:07 - Anna Maria Bigatti

- Estimated time set to 0.51 h