CoCoALib - Feature #638

Time limit: let user specify time limit for a computation

27 Oct 2014 10:45 - John Abbott

Status:	Closed	**Start date:**	27 Oct 2014
Priority:	Normal	**Due date:**	
Assignee:	John Abbott	**% Done:**	100%
Category:	New Function	**Estimated time:**	26.26 hours
Target version:	CoCoALib-0.99560	**Spent time:**	26.05 hours

Description

Robbiano asked whether it would be possible to allow a user to specify a (CPU) time limit for a computation.

Discuss!

Related issues:

Related to CoCoALib - Feature #714: Interrupt mechanism	**Closed**	**19 May 2015**
Related to CoCoALib - Feature #718: Insert calls to CheckForInterrupt	**Closed**	**21 May 2015**
Related to CoCoALib - Design #1086: New design for interrupt mechanism	**Closed**	**30 Jun 2017**
Related to CoCoALib - Slug #1181: CpuTime is costly!	**Closed**	**23 Apr 2018**

History

#1 - 27 Oct 2014 10:51 - John Abbott

- Category set to New Function

- Target version set to CoCoALib-1.0

What exactly does Robbiano want?

- setting a time limit should be accessible from CoCoA-5
- should it be a general feature, or applicable to just some fns?

What can we reasonably achieve?

- a similar feature should be in CoCoALib (which is why this issue is in CoCoALib)
- what will CoCoALib do if the time limit is exceeded? (throw an exception?)
- how accurately should we observe the limit? (if limit is 100s, but we notice after 120s, is that OK? After 200s?)
- (again) general feature, or just for certain functions?

#2 - 27 Oct 2014 11:02 - John Abbott

There is a built-in system "timer/alarm" mechanism which works via signals; I do not know much about this (incl. how portable it might be). In any case I believe that non-trivial reactions to signals have to be effected by polling (*i.e.* the program must explicitly check every so often whether the signal has been received).

Since explicit checking is needed we could simply do it directly (*i.e.* compare CpuTime() with a limit value, presumably stored in a global). I would not expect portability problems with this approach.

The accuracy of responding to a time limit depends on how often we poll (*i.e.* check whether we have exceeded the limit). However, we cannot check inside a GMP operation; so if coeffs become big, we may go considerably beyond the limit before we can poll.

Using time limits in a multithreaded environment looks to be "complicated".

#3 - 27 Oct 2014 11:08 - John Abbott

I believe it would be simplest to start with time limits being allowed only for certain functions. Amongst other things, this makes it clear when we should stop checking, and (hopefully) limits which bits of source code need to make explicit checks.

It would be nice to have a uniform syntax for specifying time limits (assuming they are offered by several functions).

Would we want to impose any other sorts of limit? (would this even be technically feasible?)

NOTE let's ignore "other sorts of limit" until we have resolved the time limit issue. Anyway I cannot think of anything else we could reasonably limit.

#4 - 27 Oct 2014 11:34 - Anna Maria Bigatti

John Abbott wrote:

> I believe it would be simplest to start with time limits being allowed only for certain functions.

I agree: I think we should start from GBasis (the most likely function to take "too long") & Co.
I don't think we need it for many functions.
In CoCoA-4 we had a function called GBasisTimeout, checking the time after each reduction (even though a reduction could take a long time, so we could miss the timeout by quite a bit).
This way there's no need for a global variable.

> It would be nice to have a uniform syntax for specifying time limits (assuming they are offered by several functions).

Now I think it could be useful (and possibly easier) to make a timeout function in CoCoA-5 for user defined functions with a time check at every line (+ inside CoCoALib functions with timeouts?)

```
Func2 := TimeOut(Func1, Sec)
```

Would that be too costly? should that be recursively set for all called functions? would that be possible?!? ..... should these lines be moved to "CoCoA-5 issues"? ;-)

#5 - 27 Oct 2014 12:39 - John Abbott

- Status changed from New to In Progress

- % Done changed from 0 to 10

I agree that GBasis computation is the obvious candidate (are there any others?).

I am unclear about whether the time limit should be specified explicitly for each relevant call, or whether there should be the possibility to say that "all GBasis computations should be limited to X seconds".

Note that GBases may be computed indirectly in many different ways: IsIn, elim, LT(I), etc.

A disadvantage of specifying explicitly for each call is that we need to add lots of new fns, namely a new "time limit" fn for each existing fn which internally computes a GBasis.

NOTE if I recall correctly, GCD may sometimes compute a GBasis (with the current implementation); would we want this to be time-limited too?

#6 - 27 Oct 2014 12:44 - John Abbott

I'm not sure about your TimeOut proposal. We should talk about details before making any decision. I'm inclined to think that it would be a lot of work (incl. significant changes to the interpreter) for a feature which is only rarely used.

#7 - 01 Sep 2015 11:17 - John Abbott

My comment in note 2 about using the system "alarm" mechanism might fit in quite well with the CheckForInterrupt mechanism. There are certainly some common elements *e.g.* having to poll to see if an external event has occurred.

#8 - 08 Jul 2017 21:31 - John Abbott

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

#9 - 15 Jul 2017 17:27 - John Abbott

- Status changed from In Progress to Resolved

- Assignee set to John Abbott

- % Done changed from 10 to 80

I have just checked in a first attempt at allowing users to impose time limits on computations in CoCoALib.

It uses the "system timer" to send a signal (SIGVTALRM) after a given number of CPU seconds; then CheckForInterrupt will detect the signal and throw a TimerInterruptReceived.

I have checked in the code with documentation and examples Not sure how to do tests.

Let me know if it works for you!

#10 - 21 Jul 2017 16:57 - John Abbott

I have modified the structure of interrupt, and now have a new version of CpuTimeLimit which does not need BOOST.
It seems to be a drop-in replacement for the version which uses the system timer.

First tests are promising; hope to check in soon.

#11 - 22 Jul 2017 15:07 - John Abbott

Checked in the new BOOST-free version.
Still have to update doc.

#12 - 08 Nov 2017 16:33 - John Abbott

- Status changed from Resolved to Closed

- Target version changed from CoCoALib-1.0 to CoCoALib-0.99560

- % Done changed from 80 to 100

- Estimated time set to 26.26 h

#13 - 23 Apr 2018 12:25 - John Abbott

- Related to Slug #1181: CpuTime is costly! added