CoCoA-5 - Feature #1129

Generic Timeout for CoCoA-5?

28 Nov 2017 15:17 - John Abbott

Status:	In Progress	**Start date:**	28 Nov 2017
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	10%
Category:	enhancing/improving	**Estimated time:**	0.00 hour
Target version:	CoCoA-5.?.?	**Spent time:**	0.65 hour

Description

CoCoALib has a generic "timeout" facility (via objects of type **CpuTimeLimit**).

Can something easy-to-use be ported into CoCoA-5?

History

#1 - 28 Nov 2017 15:22 - John Abbott

- Status changed from New to In Progress

- % Done changed from 0 to 10

In CoCoALib I have used the RAII "design pattern": the dtor for a **CpuTimeLimit** object stops the "time bomb".
So C++ scoping is the usual way of saying when to stop monitoring CPU time.

How to achieve something reasonable in CoCoA-5?

One simplistic possibility is

```
TimerVariable := CreateNewCpuTimeLimit(2.0);
RGB := ReducedGBasis(I);
TimerVariable := 0;  // to stop monitoring CPU time
```

The point being that assigning anything to TimerVariable effectively destroys the old value (which I had presumed would be a CpuTimeLimit object).
However it is a bit unnatural... perhaps it is better than nothing?

#2 - 28 Nov 2017 16:05 - Anna Maria Bigatti

John Abbott wrote:

> In CoCoALib I have used the RAII "design pattern": the dtor for a **CpuTimeLimit** object stops the "time bomb".
> So C++ scoping is the usual way of saying when to stop monitoring CPU time.
>
> How to achieve something reasonable in CoCoA-5?
>
> One simplistic possibility is
> [...]

The point being that assigning anything to TimerVariable effectively destroys the old value (which I had presumed would be a CpuTimeLimit object).

What about

```
SetTimeout(2.0);
RGB := ReducedGBasis(I);
UnsetTimeout();  // to stop monitoring CPU time
```

?

#3 - 28 Nov 2017 18:06 - John Abbott

The interface SetTimeout and UnsetTimeout is simple to understand. **BUT** it does imply use of a global variable -- not too much of a problem until CoCoA-5 becomes multi-threaded. It is a little disappointing to have to add 2 functions.

My suggestion is less natural (esp. assigning any value to TimerVar as the means to disable the timeout).

In theory my suggestion should allow "nested" timeouts so long as functions do not use TopLevel variables for storing the timer. Note that nested timeouts could happen easily if several functions which call each other have their own timeouts. Anna's suggestion would not work well with nested timeouts (at least I do not see how to make it work).