CoCoALib - Feature #974

QIR/RealRootRefine: improve behaviour if input interval has "nasty" endpoints

17 Nov 2016 19:07 - John Abbott

Status:	In Progress	**Start date:**	17 Nov 2016
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	10%
Category:	Improving	**Estimated time:**	0.00 hour
Target version:	CoCoALib-1.0	**Spent time:**	1.50 hour

Description

The fn RealRoots lets a user specify an interval in which to look for real roots.
If the initial ends points are not "nice" (*i.e.* binary fractions, and with a nice separation) then the refinement process produces needlessly ugly fractions.

Make the code aim to produce a new interval with nice endpoints (or at least one nice endpoint).

Related issues:

Related to CoCoALib - Feature #440: Port RealRoots to C++	**New**	**11 Feb 2014**
Related to CoCoALib - Feature #246: Approx QIR	**New**	**01 Oct 2012**

History

#1 - 17 Nov 2016 19:15 - John Abbott

This should really be under CoCoA-5, but I hope that RealRoots will soon be migrated to C++ (see #440).

My current thought is that the code should choose a nice interval width, and then consider subintervals whose endpoints are nicely placed. Probably the data-structure should note whether the endpoints are already nice.

Here is a quick example: suppose the true root lies close to 10, and the user gives as initial interval [1,20], successive refined intervals will be something like [1, 21/2], [23/4, 21/2], [65/8, 21/2] and so on. I hope it might be possble to get something like [8, 20], [8, 16], [8, 12], [8, 10] and so on.

The situation is worse if the original endpoints have "nasty" denominators (*e.g.* odd and coprime).

#2 - 17 Nov 2016 19:16 - John Abbott

- Related to Feature #440: Port RealRoots to C++ added

#3 - 17 Nov 2016 19:16 - John Abbott

- Related to Feature #246: Approx QIR added

#4 - 01 Mar 2021 21:25 - John Abbott

- Status changed from New to In Progress

- % Done changed from 0 to 10

I have revised impl for SimplestBinaryRatBetween so that it can accept open/closed endpoints... but it still prototypical.

Anyway, probably the most sensible approach is to take as subinterval endpoints values which are nice binary rationals.
For instance, in my example in comment 1:
original interval is [1, 20], and we want, say, about 3 interior points (which are simple binary rationals)
binary quantum 16: gives just 1 interior point, namely 16
binary quantum 8; gives 2 interior points, namely 8, 16 -- this might be a good choice
binary quantum 4: gives 5 interior points, namely 4, 8, 12, 16, 20 -- this might also be a good choice

Note that 16 is the simplest binary rat in the interval [1,20]

If we use quantum 8 then possible resulting intervals are [1, 8] or [8, 16] or [16, 20] -- the last 2 are "nice".
This will require modifying the impl somewhat.