CoCoA-5 - Design #595

rename BigIntValue, IdealValue --> INTValue, IDEALValue? or even INT, IDEAL?

23 Jul 2014 12:30 - Anna Maria Bigatti

Status:	Closed	**Start date:**	23 Jul 2014
Priority:	High	**Due date:**	
Assignee:	Anna Maria Bigatti	**% Done:**	100%
Category:	Parser/Interpreter	**Estimated time:**	10.00 hours
Target version:	CoCoA-5.1.1 Seoul14	**Spent time:**	11.00 hours

Description

In the interpreter (AST.H) we have the definition of the classes BlahValue, for example
BigIntValue, IdealValue.
Philosophically, these are the internal representations of the CoCoA-5 objects of type INT and IDEAL (which are represented by CoCoALib object of type BigInt and ideal).
There is a BlahValue for each CoCoA-5 type BLAH (and a few more).
So, should they be called INTValue, IDEALValue?
or even INT, IDEAL?

2014-08 decision was for INT, IDEAL,..

History

#1 - 23 Jul 2014 12:32 - Anna Maria Bigatti

- Subject changed from rename BigIntValue, IdealValue --> INTValue, IDEALValue? to rename BigIntValue, IdealValue --> INTValue, IDEALValue? or even INT, IDEAL?

#2 - 23 Jul 2014 12:38 - Anna Maria Bigatti

This is how some lines in BuiltinOneLiners would look

```
DECLARE_COCOALIB_FUNCTION1(factorial, INT)
DECLARE_COCOALIB_FUNCTION2(NewQuotientRing, RING, IDEAL)
DECLARE_COCOALIBFORC5_FUNCTION2(GetCol, MAT, INT)
```

and in BuiltinFunctions

```
DECLARE_STD_BUILTIN_FUNCTION(LT, 1) { // AMB
    int which;
    intrusive_ptr<RightValue> v = runtimeEnv->evalArgAsT1orT2orT3orT4<RINGELEM, MODULEELEM, IDEAL, MODULE>(ARG
(0), which);
  switch (which) {
  case 1: return Value::from(LT_forC5(PtrCastRINGELEM(v)));
  case 2: return Value::from(LT_forC5(PtrCastMODULEELEM(v)));
  case 3: return Value::from(LT(PtrCastIDEAL(v)));
  default:return Value::from(LT(PtrCastMODULE(v)));
  }
}
END_STD_BUILTIN_FUNCTION
```

(I'm thinking of my talk in Seoul, that's why I'm so concerned ;-)

#3 - 24 Jul 2014 21:19 - John Abbott

- Status changed from New to In Progress

- Priority changed from Normal to High

- % Done changed from 0 to 10

- Estimated time set to 7.70 h

Certainly the excerpt from BuiltinOneLiners is very clear.

We have already spoken about the use of default in the switch statement.

I'm not entirely convinced by the names PtrCastIDEAL etc.
It is true that the arg is a "pointer" (boost::intrusive_ptr), but the result is a reference (yet the name suggests to me that the result should be a pointer). Also the result is (a ref to) a CoCoALib type, but the capitalized names suggest CoCoA-5 types (it is true that the arg is effectively an object of the named CoCoA-5 type).

Here are some other possible names:
RefToBigInt
ViewAsBigInt
GetBigInt
GetRefBigInt

#4 - 25 Jul 2014 08:47 - Anna Maria Bigatti

- Assignee set to Anna Maria Bigatti

- % Done changed from 10 to 30

I started the change: there is no name clash problem with the class INT :-)
... at least with gcc :-/
Anyway: in case some problem arise in the future we just need to replace INT (or STRING, TYPE) with another name in

```
AST.H
Interpreter.[CH]
BuiltIn*.[CH]
```

#5 - 25 Jul 2014 08:51 - Anna Maria Bigatti

John Abbott wrote:

> I'm not entirely convinced by the names PtrCastIDEAL etc.
> It is true that the arg is a "pointer" (boost::intrusive_ptr), but the result is a reference (yet the name suggests to me that the result should be a pointer). Also the result is (a ref to) a CoCoALib type, but the capitalized names suggest CoCoA-5 types (it is true that the arg is effectively an object of the named CoCoA-5 type).

true: it is clearer with the return type.

> Here are some other possible names:
> RefToBigInt

I like that! even shorter than PtrCastBigInt

#6 - 25 Jul 2014 09:11 - Anna Maria Bigatti

First pass is mostly done (BigIntValue --> INT, and similar).

I'm a bit undecided about a few other types like OSTREAM (because it's used also in Lexer)
There is a CppOstreamValue I can't understand right now...

#7 - 25 Jul 2014 12:53 - John Abbott

Another possible name is **UnwrapBigInt**. I think it conveys the idea that the BigInt is hidden inside the objects passed as argument, and that the result is a reference to that same object (rather than a copy).

I'm tempted to suggest use of templates: *i.e.* the function would actually be called **unwrap<BigInt>**. It is almost an abuse of templates because each specific function has to be defined separately (at least with the current design a generic template definition is not possible). But it somehow "feels right" that the name of the type be given as a template parameter... It is probably also a bit easier to read.

What do you think?

#8 - 25 Jul 2014 16:16 - Anna Maria Bigatti

Now all switch following EvalArgAsT1or... have a default case giving and error.
This is good for maintenance: if we add some extra type and forget to write the appropriate case, it will give a meaningful error.

#9 - 25 Jul 2014 19:50 - Anna Maria Bigatti

added templates for RefTo<..>.

#10 - 28 Jul 2014 10:55 - Anna Maria Bigatti

- % Done changed from 30 to 80

- Estimated time changed from 7.70 h to 10.00 h

Changed also also PtrCast into RefTo<..>
cvs-ed

#11 - 04 Sep 2014 12:32 - John Abbott

- Status changed from In Progress to Closed

- % Done changed from 80 to 100