Subject: Re: [DISCUSSION] Add 'complex' datatype, to Value too
Posted by kohait00 on Sun, 26 Jun 2011 14:05:13 GMT
View Forum Message <> Reply to Message

ok this all doesnt help..we got a semantic problem here.

complex can be constructed both from double and anther complex.
same applies for operator=, it can assign a double or a complex.

now, for Value support (and Nuller) we have both operator double() and operator cdouble() implicit conversion, that applies here.

so the compiler could construct a cdouble from a Value (or Nuller) in 2 paths, converting it to double or cdouble. and it does not know which one to prefer.

cdouble x = val.operator double();
cdouble x = val.operator cdouble();

both work, same for Null.

thats due to the implicit handling of double as a sole real part in complex context in std::complex. so there is no way around that. not even deriving from std::complex, unless we spare out the complex(double) ctor and operator=(double).

and it only touches Value conversion stuff. one could leave out the Value support, but it's bad too. so at least to have the Value support like that (dealing with explicit conversion access in this special context) is an advantage, at least the user could know what to call if the compiler doesnt.

RESULT: though it's 'ugly' to explicitly call Null.operator cdouble() and v.operator cdouble(), it's a reasonable price to pay for a common std::complex usage which we dont need to code oursselves anymore. Nuller.operator cdouble() even could be left out, so it only would apply to Value.operator cdouble() to call. but i'd keep it the same semantic, Null.operator cdouble() as well.

is it worth it, that's the final question? everything else works..