T:Basetype
ET:ErrorTerm<T>
AC:AffineCombination<T, ET<T> >
IV:Intervaltype

<<concept>>

ArithmeticKernel

+base_t: typedef = T
+error_t: typedef = ET<T>
+ac_t: typedef = AC<T, ET>
+aerror_t: typedef = yalaa::details::ArithmeticError<T>
+self_t: typedef = ArithmeticKernel<T, ET, AC>

+add(s1:ac_t,s2:base_ref_t): aerror_t
+div(d1:ac_t*,d2:base_ref_t): aerror_t
+scale(p1:AffineComination*,p2:base_ref_t): aerror_t
+add(s1:ac_t*,s2:const ac_t&): aerror_t
+sub(m:ac_t*,s:const ac_t&): aerror_t
+mul(p1:ac_t*,p2:const ac_t&): aerror_t
+div(d1:ac_t*,d2:const ac_t&): aerror_t
+sqr(ac:ac_t*): aerror_t
+sqrt(ac:ac_t*): aerror_t
+pow(ac:ac_t*,n:int): aerror_t
+pow(ac:ac_t*,exp:base_ref_t): aerror_t
+pow(ac:ac_t*,exp:const ac_t&): aerror_t
+exp(ac:ac_t*): aerror_t
+ln(ac:ac_t*): aerror_t
+log(ac:ac_t*): aerror_t
+sin(ac:ac_t*): aerror_t
+cos(ac:ac_t*): aerror_t
+tan(ac:ac_t*): aerror_t
+affine(ac:ac_t*,scale:base_ref_t,add:base_ref_t): base_t
+affine(ac1:ac_t*,ac2:ac_t,scale1:base_ref_t,
scale2:base_ref_t,add:base_ref_t)