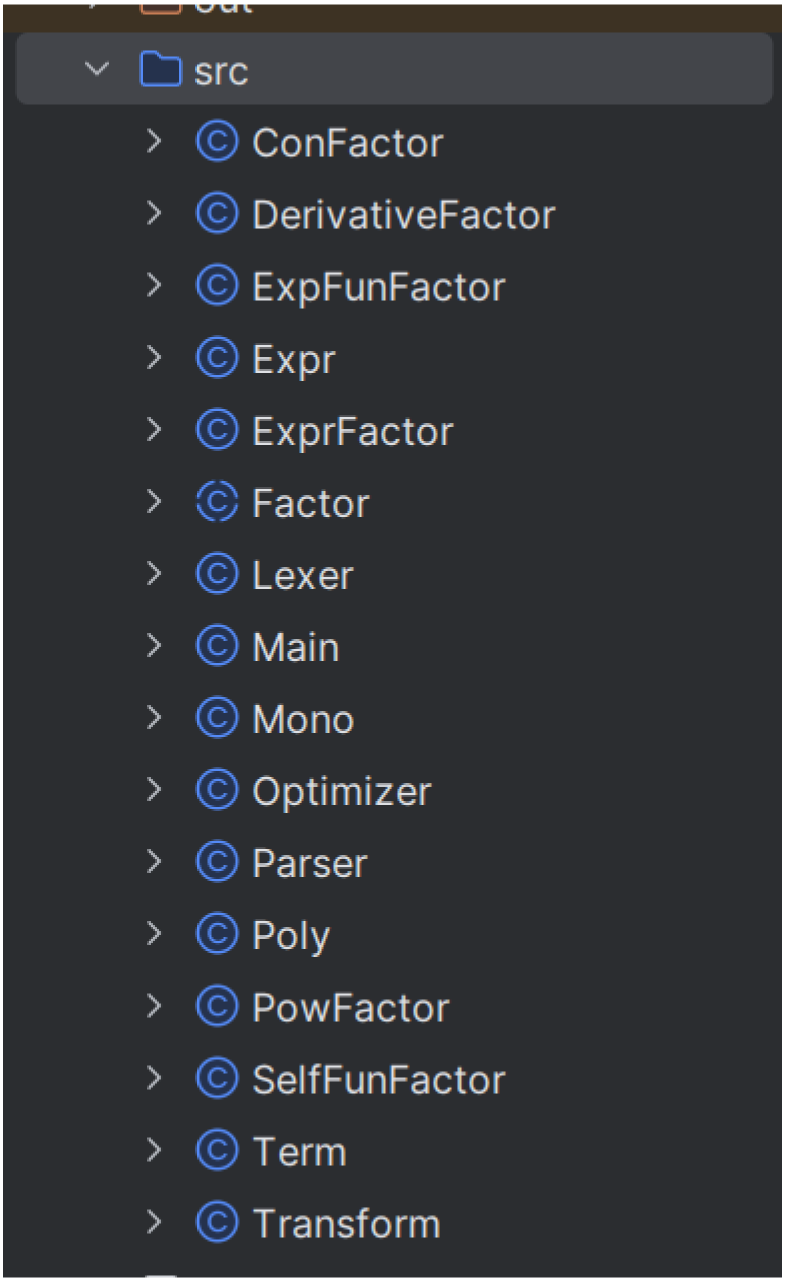
hw_3 作业完成思路分享

前言

本次作业与上次相比，需要额外处理两件事：用已定义函数定义新函数+求导。以下是本次作业所有文件，其中`DerivativeFactor`为唯一新增的类。



用已定义函数定义新函数

对于用已定义函数定义新函数而言，由于我是采取的hyggge博客中说到的方法，所以不需要任何额外操作就可以实现这个功能。同学们可以参考第二次作业讨论区于耀博同学的解题方法，我认为写的已经足够详细。

求导

对于求导的话，我首先新建了一个`DerivativeFactor`类，负责管理**求导因子**，在该类中用`Expr base`存出`dx(......)`括号中的表达式。

做完前一步建类工作后，我在`Expr`,`Term`以及各种`Factor`的各个子类中都实现一个`toDerivative()`方法，且均返回`Poly`类型的对象。这个方法的作用是返回由该部分的导数而构成的`Poly`。

$$\mathrm{I}.\text{当}f(x)=c\text{（}c\text{为常数）时，}f'(x)=0$$

$$\mathrm{II}.\text{当}f(x)=x^n\text{（}n\neq 0\text{）时，}f'(x)=nx^{n-1}$$

$$\mathrm{III}.\text{当}f(x)=exp(x)\text{时，}f'(x)=exp(x)$$

$$\mathrm{V}.\text{链式法则：}[f(g(x))]'=f'(g(x))g'(x)$$

$$\mathrm{VI}.\text{乘法法则：}[f(x)g(x)]'=f'(x)g(x)+f(x)g'(x)$$

接下来我们就可以用课程组给我们的求导法则来写各个类的`toDerivative`了。

- 对于表达式，直接将各项用`toDerivative()`返回的`Poly`类型对象相加即可；
- 对于项,则应用导数的乘法法测即可；
- 对于常数因子，其求导后得到的值是0，则将0转化为`Poly`即可；
- 对于幂函数因子，应用图中方法2即可；
- 对于表达式因子和指数因子的处理比较复杂，但是也无非是链式法则和图中方法2的结合罢了。

这些处理方法再复杂也不过是对于上图公式的应用罢了，本身没有什么难度，值得一提的是如何求`DerivativeFactor`的导数，也就是遇到`dx(DerivativeFactor)`，其中`DerivativeFactor`实际上是`dx(Expr)`，这里实际上需要求`Expr`的二阶导，那么显然地我们应当先求内层的导数，然后再求外层的导数。

但是我们可以发现求内层导后会返回一个`Poly`类型对象，而我们求导的方法本身又无法处理`Poly`对象。所以我在这里先将获得的`Poly`对象转化成字符串（实际上就是将这个`Poly`当做最终的`Poly`，走一遍优化和输出的流程即可，非常简便），然后再对这个字符串（实际上已经没有`dx`了）进行词法分析、语法分析，也就是重新走一遍之前的过程，得到一个新的`Expr`，再对这个`Expr`求外层导数即可。