数字生命

罗瑶光

浏阳德塔软件开发有限公司

数字生命进化主线

- 德塔开源大数据项目集的诞生
- 元基 AOPM 初始因子的由来
- 元基 VPCS 初始因子的由来
- 元基 IDUQ 初始因子的由来
- 元基 TXHF 初始因子的由来
- DCPE THOS MAXF VIUQ 元基进制编码
- 大数据项目集所有函数进行元基编码分类索引
- 索引序列化 映射 函数接口
- 人类语言 引导 索引 进行搜索映射 计算
- 语言引导 进行 序列化编码 生成 可遗传 基因

德塔大数据项目的诞生

- 德塔大数据 开源项目集 最早是罗瑶光先生为其父亲设计一个 中医的 药材搜索软件， 逐步演化而来。
- 这个软件目前包含搜索，排序，分词，DNN，视觉，听觉，预测，统计，变换，脚本，ETL等数据基础计算组件集。目前已有一个名字叫养疗经。
- 作者不断的把软件的公共函数提取出来做成API。作者希望能将其设计成自主分析计算遗传的生命API，取名为华瑞集。
- 自从2018年10月开始，作者开始潜心探索数字生命的模型和编码框架。

元基 AOPM的由来

作者在 2008年大学 比较系统的学习了《软件工程》课程，在很多项目中进行实践，发现软件的瀑布设计模式中 （采集-细化-分析-操作-编码-测试-运维-优化），可以进行分类， 如细化分析用分析 操作替代，采集编码操作可以用操作替代，运维测试可以用处理替代，运维优化可以用 操作管理替代，于是浓缩成分析analysis-操作operation-处理process-管理management，4个子集。

元基 VPCS的由来

- 作者第一次接触MVC-模型Model-观测View-控制Controller是在使用Java Spring架构2009年，后来同年涉及MPC-模型Model-处理Process-控制Controller的手机线程调度架构，于是将MVC中C的控制Control和执行Execute分离，将MPC中P拆分融入进入执行Execute和静态StaticData变成Model-Controller-Execute-StaticData,再逐步进化成将M拆分融入进入Vision子集管理和Hall全局管理，变成HVPCS，然后去重P得到HVECS

元基 IDUQ的由来

- 作者在2003~2013的大学时代，广泛的学习了《数据库概论》，《数据库管理》《数据库原理》，《数据库系统》，《数据挖掘》，《专家系统》，《离散数学》，《数字逻辑》等专业课程。发现数据的操纵计算模式主要为四个增-删-改-查，于是定义为Increment-Decrement-Update-Check，将C-check去重得到 IDUQ-Increment-Decrement-Update-Query，变成Accumulator 计算模型。

元基 TXHF的由来

- 作者将AOPM-H-VECS-IDUQ 编码 模拟人类DNA已知的ACGTI 5个嘌呤嘧啶基元 按语义推导匹配，竟然吻合， 还顺便推导出一整套语义肽展公式集合PDE -PDn-Extension。然后通过肽展公式进行生化解码推导，把TXF-触发Trigger-探索Xplore-全Full， 3个 元基给语义定义了。

DCPE THOS MAXF VIUQ 元基进制编码

- 自从十六元基 和 肽展公式 的逐渐规范化，作者开始将其应用在真实环境中，逐步将其进行索引化，工程化，分类以及各种观测，模拟数字逻辑计算进制规则，得到很多成果。于是有信心开始探索数字生命的奥秘。

索引序列化 映射 函数接口

- 作者将十六元基，中的 前三组稳定元基组AOPM-VECS-IDUQ 进行 模拟染色体分类，分出了24个类{A-VECS,A-IDUQ, O-VECS P-VECS……I-AOPM,I-VECS,…Q-VECS}，然后将养疗经 华瑞集的所有函数进行static静态接口化，然后将这些接口分类并到24组中。然后统一写一个接口搜索索引来调度这24组接口集。

人类语言 引导 索引 进行搜索计算

- 于是，作者开始设计人类的语言进行处理变成格式化脚本命令，然后驱动这个索引按先后顺序来操作24组接口集 中的函数，并进行序列记录。

语言引导 进行 序列化编码 生成 可遗传 基因

- 这个先后顺序的索引链 一旦编码，作者认为 这是一种 数据在 处理任务中 获得的 认知层次的 可遗传基因。
- 这个基因链 可以编码解码，可以遗传配对，可以裁剪优化，可以存储计算。
- 同时也是一种具有解决问题能力的劳动的方案记录。

结束

真实作品 已经Git 开源

1 ppt 《女娲计算 2021》

2 pdf 《DNA元基催化与肽计算第三修订版V 039010912》

3 jar 《BloomChromosome_V19001_20211227.jar》

实例，单一计算

```
Map<String, Object> output= new HashMap<>();
String[] 传参因子= new String[2];
Map<String, Object> inputValue= new HashMap<>();
double[] doubles= new double[5];
doubles[0]= 2.2222262;
doubles[1]= 3.2226222;
doubles[2]= 6.2622222;
doubles[3]= 4.6226222;
doubles[4]= 1.2222226;
double dou= 2.22;
传参因子[0]= "input";  传参因子[1]= "rank";
inputValue.put(传参因子[0], doubles);
inputValue.put(传参因子[1], dou);
output.put("传参因子", 传参因子);
output.put("inputValues", inputValue);
strings[2]= "执行 U_AOPM 下 min_v 接口, 参数是 传参因子";
//... StaticRootMap.tinShellV003(strings, output);    输入结果为 1.2222226
```

实例，轮循计算

- double[] doubles= new double[5];
- doubles[0]= 2.2222262;
- doubles[1]= 3.2226222;
- doubles[2]= 6.2622222;
- doubles[3]= 4.6226222;
- doubles[4]= 1.2222226;
- double dou= 2.22;
- 传参因子[0]= "input";
- 传参因子[1]= "scale";
- inputValue.put(传参因子[0], doubles);
- inputValue.put(传参因子[1], dou);
- output.put("传参因子", 传参因子);
- output.put("inputValues", inputValue);
- strings[2]= "执行 U_AOPM 下 median1d 接口, 参数是 传参因子";
- strings[3]= "执行 U_AOPM 下 fengTong1 接口, 参数是 过程因子";
- StaticRootMap.tinShellV003(strings, output);