第七章 类人DNA与神经元基于催化算子映射编码方式 原文 的英文部分， 图片在如下链接。或原书第七章中。

**The Initons Catalytic Reflection Between Humanoid DNA and Nero Cell**

Yaoguang Luo, Rongwu Luo

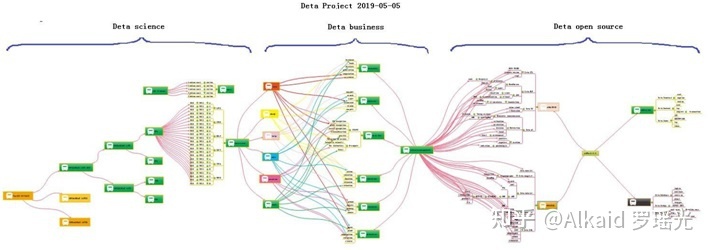
Keywords: VPCS, AOPM, IDUC, Nero, Artificial, Decoder, Medical, Paralling, Computing,

Humanoid, ETL, Parser, Data Mining

**Outlook:**VPCS architecture is not the end. Absolutely, At least at this paper, I will make an implementation in five sections: DETA humanoid cognition, DETA Medical Business backend logic, DETA Catalytic computing, DETA Findinginitions, DETA DNA decoding.Above all I also will spend more and more words in my DETA DNA Law of IDUC. And itsapplications in the real world. ok next step as below.

**1 DETA humanoid cognition**

Since the inition of the DETA OSS, there has a lot of questions where based on the humanoid DNA catalytic computing, I have been working on this domain for a long time. Absolutely also, I have got a lot of flashing points here, for example AI, still remember the first time I touch the cognition this verbal at CLU Dr Renhat’s class about cogs PU computing and cognition quality Sonar test where in Folsom Intel, I know that cog-work is a trending task in my life. Even though I need do a lot of basic foundations at AI, but I insist to now, from the code where between normalization and duplication. I have been thinking how to make a contraction and distinction AI logic with human and humanoid. Ok, PLAN A starts as below. Code a problems solution software like a way of YaoGuang. Luo's cog-style life.



Look at this PIC, smartly, build basic foundations first, then create more business software where based on this foundations, and finally swap to a humanoid model. Many many times I hope the model could be an Immortality.

**1.1 DETA humanoid cognition history,**

In the past, knowledge of the world could be parsed by five sections, the world's cognitive way, philosophers and scientists liked to describe objects with five senses (touch, taste, hearing, smell and vision). Species sense can make creatures in order to adapt to the environment well, understand the environment, and think about ways to protect themselves in dangerous environments. Thesemethods are implemented in a variety of ways, and their execution logic is also varied, but the causes and results are clear about a basic point. Better adapt to the environment. During the process of designing the YangLiaoJing, the author has well integrated the bionic technologies of voice, text, association and visual media, and has been optimizing them to gradually form a comprehensive intelligent YangLiaoJing system.

**1.2 DETA humanoid cognition development,**

In order to better adapt to the environment, human beings began to create characters, invent tools and improve their cognitive ability, from the ignorance of slaves to the open and compatible world, from the Iron Age to the current nano-chip technology. In the river history of 5000 years, human beings seem to have evolved aimlessly. Through these phenomena, the essence can be easily discovered. Better adapt to and transform the environment. The best arguments to improve the cognitive ability of environmental things are the research and development of basic science and technology and systematic induction, Marconi's wireless telegraph, Zu Chongzhi's pi, Darwin's origin of species, code of Hammurabi, and countless outstanding scientists, thinkers, inventors and philosophers in history. These people seem to be shining stars in the night sky. Gradually, Intelligent creatures begin to have enough ability to look up into space and explore the mysteries of the universe, and this enough ability is the basic ability to improve the cognition of things, which is very important. The HuaRuiJi system of DETA Company draws lessons from the systematic induction method of human instinct and dialectically treats medical diseases with medical textbooks as the cognitive basis, which is in line with the embodiment of scientific development.

**1.3 DETA humanoid cognition application,**

There are many applications of humanoid cognition in social science, and there are many excellent arguments here, such as auxiliary auditory system, big data reasoning system, weather forecasting system and criminal investigation database system, which undoubtedly proves that cognitive model has greatly improved the adaptability of human beings to the environment. To the ability to gradually transform the environment from part to whole. There are many outstanding arguments here, from the ancient Dayu flood control, to the renovation of Emperor Yangdi's Canal, (当前政治经济实体实例已经过滤), from artificial rainfall to artificial islands. The argument is very clear, and the cognitive ability of things comes from the accumulation of basic science and technology. These basic technologies gradually form a system. On top of it are a wide range of scientific and technological commodity applications, (当前政治经济实体实例已经过滤), giant hydro-power station of Three Gorges of Yangtze River to improve Geographical environment, etc. There are too many. The evolution of human wisdom gradually forms a clear route, and improving basic science and technology and cognitive ability complement each other. These abilities all come from thinking about things. Then form a solution, and finally implement it. This process is summarized as the process of analysis, operation, processing and management. The life cycle of software engineering is well explained here. Analysis A, Operation O, Processing P, Management M, use simple words to describe that even if unknown data are collected and analyzed, and then things are operated, the solutions to various difficulties encountered in the operation process are implemented. Finally, maintain and manage these implementation experiences. The back-end computing mode and system life cycle of DETA have gradually condensed from the earliest collection, analysis, operation, sorting, coding, running, debugging and maintenance to the module modes of Analysis A, Operation O, Processing P and Management M, such as DETA word segmentation, DETA DNN mind reading, etc. Now ETL of DETA is ready to go in this direction. The author designed a paper last year to describe the application mechanism of AOPM as follows:

**AOPM Open Source System On SDLC Theory**

Mr. Yaoguang. Luo

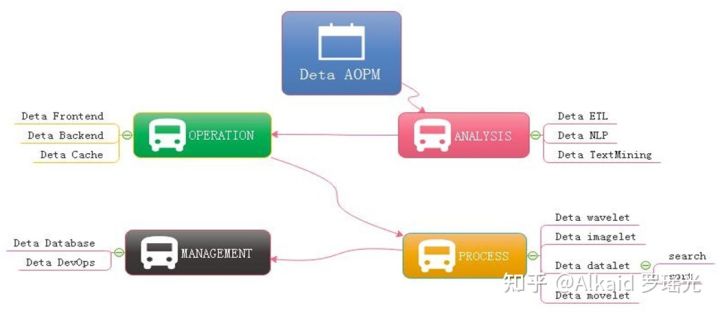
Outline: Mr. Xuesen. Qian once said: Science Is A Titan System, as an open source software conception. this topic implements a software interaction theory of SDLC for Analysis, Operation, Process and Management— AOPM. Also, this is a tiny paper where easy to show more idyllic landscapes of using DETA open source projects. Not only for web system, also for mobile and desktop platform. The final goal are makes complex project to simple. Ok let go and the next steps.

Keywords: SDLC, AOPM, VPCS, WEB, Concurrent, Open Source, Interaction, Management, Automation

Introductions Recently my colleagues take more care on the SDLC evolution of open source software engineering, for each project they undertake on where it cost a lot of times, that’s for my job, continuing found out a high effect, simple and clear theory of SDLC what be my main task now. after imagination and logic recursion, the key is an optimization of ordinary SDLC such as water fall. First time for makes an introduction to waterfall of SDLC? The author’s explanation likes sequence linked list of component nodes. With DETA projects here contains four aspects at Figure1-1. And my explanation of open source as belows.

**Topic: Ten Definition of The Open Source, OSS Book Reading Note**

In this paper, through a premise: the contrast between the copyright and the contract. the Author talks a comprehensive introduction to the definition of the open source code. The role of the open source licenses, which is to allow the work permit under the non-exclusive business. Not only does it mean that the source code was visited by the public user, and also meets another 10 conditions as follows. The first point: the open source software allows the free reusable distribution. The license must not restrict that any party sell or give away the software. At the same time, it can't get the sold fees and other fees for this software. The second point: the program must include the full of source code. The license does not allow that getting the source code from any specific forms of the production. The license assures that no one can intentionally to confuse the source code. At the same time, the users have the right to access to the source code under this license. The third point: which talks about the rights of the derivative work. The license must allow the work-modification and the new-work-derivation . those new's are published under the same license. The fourth point: the integrity of the source code. Licenses and the integrity of permits,which may limit the distribution of the form of the modified source code. The fifth point: license does not discriminate against any specific groups and individuals. The sixth point: license does not limit the use way of any particular field scheme. At the same time, the license can't limit the use way's flexibility and reliability. The seventh point: the distribution of the license. Distribution solutions do not need additional license.The eighth point: the license must not specific to the product. The redistribution of the software does not dependent on the program. The ninth point: license may not restrict other software. This license may not restrict the publish of the software. The distribute software will be built by using open source. The end point: license rights is neutral. So, it effective limits that the freedom of the code transmission. In other words, it provides the preventive measures.



**Figure 1-1 AOPM Applications with SDLC Evolutions**

Last year I was asked by so many engineers, almost the same question: how have you done so many projects during the year of 2018~2019 My answer is absolutely: connection. Always, with connection, I got lots of fantasy inspirations on the projects where I undertook. My projects all are lower basic technical factors, with connections, what support me the necessary energy for continuing development on my projects. What means connection? Be an internal union bridge between my projects. For example DETA NLP and DETA ETL, they both have the same attributes such as AI, Analysis and Data etc, with this connections, my tasks became more dynamically. Every time before I made a decision of priority levels of my projects, I thought the connection first, DETA projects totally can be separated into three dimensions. Front end Back end and Storage, as the Figure 1-2, the connection between DETA projects is WEB AI, now is a Bazaar requirement, but we will easy to make estimation of it’s future, toward to Cathedral.

**Topic: Cathedral and the Bazaar, OSS Book Reading Note**

Cathedral and the Bazaar, this article has a profound implication, the author is a computer scientist with extensive experience. We can say that he is one of the early code and program contributors in the UNIX system. This article describes the Linux development with the revolutionary road, as the process from the bazaar to the cathedral. First, the author tells the contrast between UNIX and Linux: now UNIX is still popular around the world. Its rigorous structure and contribution to science, let it is proud of the same dignity as a church. Linux looks like a noisy bazaar, the code work in various countries around the world, to solve their own problems and arguing in the forums and communities. Like a bazaar. Then, author points an internal factors to get an in-depth discussion: UNIX reason why it has the church's authority, because its development has always been tailor-made by the world's most senior and most eminent researchers and software scientists. Although the discussion, because of the nature of the project-oriented, so that UNIX has been applied still to today. Even of the unreasonable original design, through decades of use, engineers have become accustomed to this experience now, there fore, we are called transcendental. which makes UNIX feels like a cathedral. The birth of the Linux was different, survival in an all-spittle environment. Every update, are implemented in controversial circumstances. The crowd here, are huge number of scientists, or writers,or code workers or merchants, their common ideal is that make Linux development meets the needs of all groups. Similar a huge bazaar. The author commenced a leno-vo, a conclusion that Linux will eventually beat UNIX, UNIX gets the range of fresh blood is less than the Linux' s, also the number of the UNIX team members is less than the Linux' s. UNIX customers and employees are aging. But Linux development more in line with the user of the needs. Its own development is to establish a relationship on this demand and requirement. Linux is young now. Summary, UNIX and Linux development option is the two kinds of very different road. These processes and methods to determine the fate of the two kinds of software development. Of more optimistic about Linux because it is better adapted to the environment.

At figure 1-2, DETA open source main based on AI domain, it already formed as an ecology system, go ahead to the application, thanks.

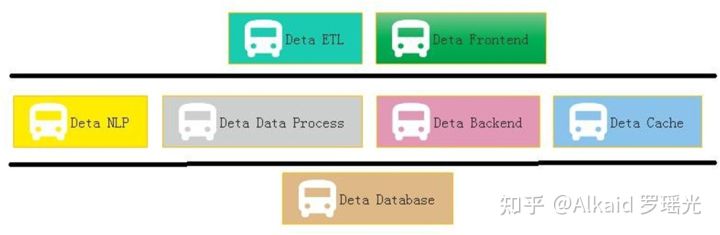


Figure 1-2 Sections of DETA Projects Group Applications One question is my friend asked me why does DETA support the e-commence logic? Definitely! Please see the Figure 1-3, this is a classic horizontal deployment sample of the real word. Alibaba, Amazon, Ebay and JD etc, all based on this technology, instead of Spring, DETA can be the next generation of technology.

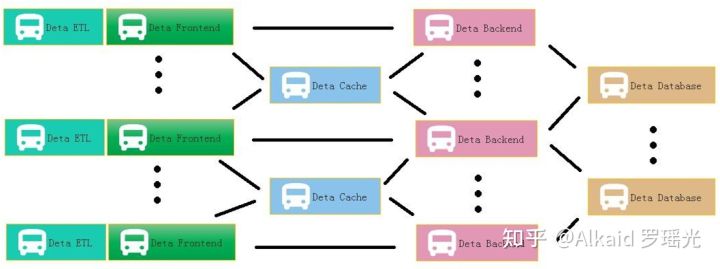


Figure 1-3 DETA WEB Projects System At Figure 1.4 is a real sample for web Devops by using DETA Open Source.

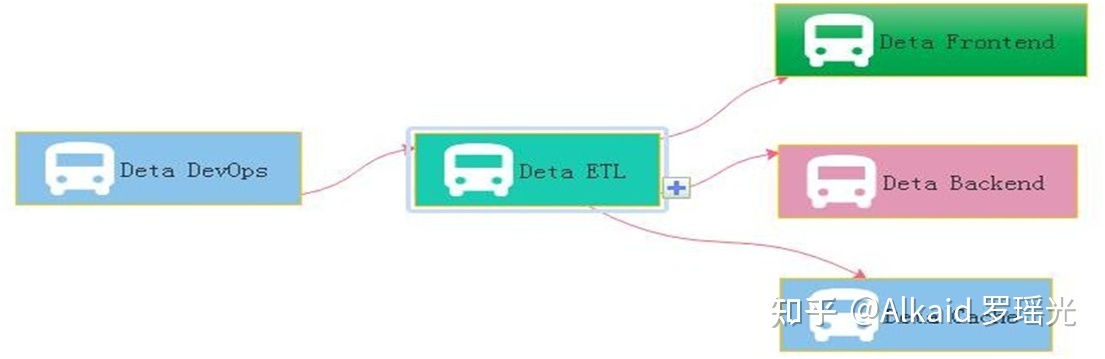


Figure 1-4 DETA DevOps Projects System

**2 DETA Business back end logic**

Before 2010, the author systematically contacted the mechanism of analyzing A, operating O, processing P and managing M in the learning process. After graduation, he had the opportunity to deal with the business logic corresponding to these things through programming in some software companies in the society. From the research of MP6 mail system (当前政治经济实体实例已经过滤), The Bluetooth group advertisement machine to (当前政治经济实体实例已经过滤), from the e-commerce back-end calculation(当前政治经济实体实例已经过滤)to the global hotel reservation (当前政治经济实体实例已经过滤), the author has been thinking about how wonderful it would be if the front-end system could give wisdom like human beings. So the bud in the author's heart began to take root, and he was confident to design a set of architecture system with humanoid wisdom to meet the rapid development of business intelligence applications.

**2.1 DETA Business backend logic history,**

The first contact with the Front end and Back end separation was in 2004, when the author first published a website in Liuyang city by using the 7week platform. The website was a second-level domain name, using a third-party server, even though the concepts of front-end and back-end were ignorant at that time. The author first contacted MVC architecture in Shanghai Fan Teng Information Technology Company. At that time, the feeling was that MVC could solve all kinds of business logic. In the same year, the author first came into contact with MVP to do multi-thread Bluetooth big file project, and felt that MVP seemed to make the architecture handle the problem of concurrent computing well. From 2014 to 2017, the author worked almost with business logic corresponding to various MVC architectures, such as Spring, Martini, etc. The author thinks that gives MVC an intelligence urgently.

**2.2 DETA Business backend logic development,**

Thanks to my father, in 2018, he told me to design a pharmacy-assisted search software according to the concept of Chinese medicine, so he began to design Huaruiji Medical Big Data System. At that time, I thought spring boot, mysql were too heavy. If the database rest handshake system of socket stream was designed according to CGI, many problems could be solved easily. So I began to analyze, operate and deal with the problems. Gradually found some irreplaceable primitives, such as S static data, V visionary observation model, P procedure registration mode, C control unit, etc. It would be wonderful if we could redesign a set of architectures for these primitives. The PC separation mode here comes from an IOC doctoral design paper in Spring in 2015. Thanks here, I integrated MV into V observation model, and then took out the corresponding static data of M and function S. This VPCS structure choots me at present.With regard to the excessive description of VPCS, I can take a previous note as follows: VPC architecture programming thought, software programming for many years, accumulated some thoughts on program realization. Through the certification of Darwin's theory of evolution, an effective VPC programming concept is elaborated based on the neutral coupling of MVC+MVP. V is an observer model, similar to storage object and observation model. P is the processor, which handles the registration interface. C is the control machine, which describes and classifies the registration interface. S static control machine, why use static control machine, advantages: 1. Because of the separation of PC, the functions of C mode are inherited through abstract virtual functions, interfaces inherit interfaces, interfaces are uniformly registered, and calls are extremely discrete, thus achieving the efficiency of high-speed concurrent iteration. 2. Realize EI separation and skip IOC scanning. 3: P is responsible for reference and description, and C can carry out various functional operations through descriptions of multiple P. Mapping control technology ensures thread safety and stability. 4: V stores each single case class to ensure low data redundancy and unified recovery.

**2.3 DETA Business backend logic application,**

In the whole year of 2019, the VPCS back-end engine gradually formed some standardized functions and papers, which were applied to the front-end, back-end, cache, database and other subsystems of DETA. My evaluation of them is that they are lightweight and extensible. VPCS is gradually integrated into the works of Yangliaojing and Huaruiji. Of course, there are many shortcomings, the biggest one is that they do not repair themselves. Although I designed the sleeper and hall keeper mechanisms, these mechanisms are only the corresponding business logic units that I complete through decision trees, not humanoid evolutionary thinking. At least, I don't think they are humanoid intelligence. To be precise, at present, they are only artificial intelligence, a kind of artificial intelligence logic corresponding to AOPM and VPCS, but not the humanoid evolutionary intelligence logic that I want. So I started to explore humanoid computing again. About the application principle description of VPCS, the author designed a paper as below:

**VPCS Backend Theory And Its Application**

Mr. Yaoguang. Luo

Outline: due to the development of the software acquisition and definition in what we use the code theory always in messy and unforeseeable status. A new method of the coding style like VPCS that will show in this topic paper, feel free to resonate with my imagination of the portrait—VPCS(Vision, Process, Controller, Sets) theory, fun yet? Not only this paper will gazer a big point how we show the onstructions of the VPCS, you guys also sure to get lots of idyllic landscapes of the coding sections. While you got lots of the illness codes at the so messy fungus projects, I guess at this paper out where you are finding anxiously. Let’s catch more opportunity about how does the VPCS working, executing and scheduling in our software project and make the software fast, fast and safe! lets go, So the key words as below:

Quantum Sets, Concurrent Consumer, Vision, Scheduler, Threads, Surf.

Introductions

Let see the verbal keys, the first time you …, okay, get any sense? Sure, this paper is not talking about the human careers, truly about software, as a human, if you got my points, yes, cool! Make any sense? Let’s see the landscape as below figure 1-1.

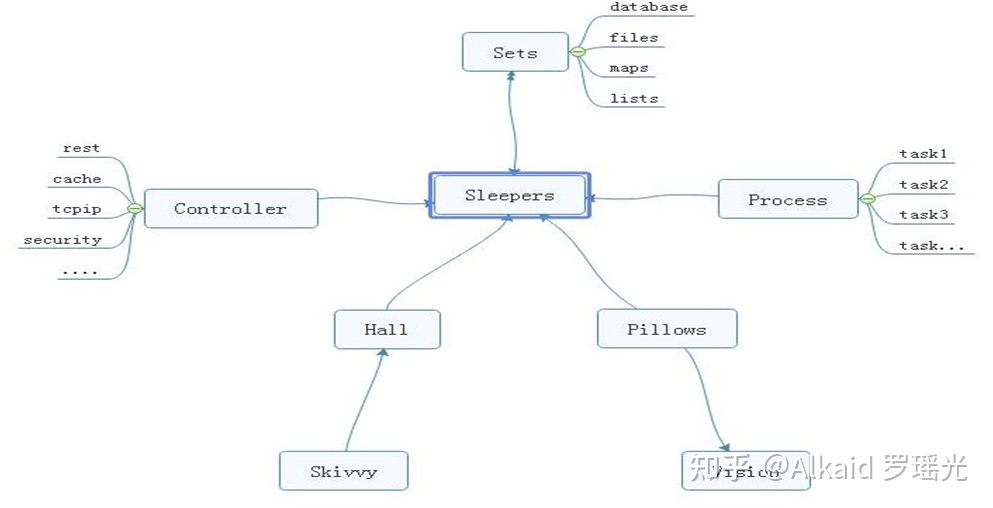


Figure 1-1 VPCS STAR MODEL

From the ordinary software development architecture, always like a factory model, for instance, controller, transaction delegate, web service, job bean, data DAO, like that of traditional back end or front end coding style, but, compare now the seamless clients services system, those model more and more not suitable for us for the project application, at least in the light level, multitasks, satellite boots projects system, if we choice the factory model, you will feel so heavy. But the big conflict problem is where the factory model was used in all and all bazaar companies. Even more CTOS that I met before often complaining about the reference room likes that “we need one server for database system, one more for cache system, one more for front end, one more, for backend, one more….”after that what do you think? My lord…Finding a new method of how to integrate the sets about the micro satellites service in the same sever, and make them small, lightly and faster for the commence service, now become a fatal topic. Which can be a pretty warm-up for where I make an explanation for VPCS. The VPCS model, only includes four aspects. Vision, Process, Controller, Sets, and those factors makes an interactions in the sleeper containers. Let talk about the definition of the sleepers. From the software engineering domain, the sleepers are more like an identified thread person. Who can make a lot of fantasy dream in a Hall, what means a dream? Dream is a requirement what the consumer really needs to finished. But here the dream can be separated out more tasks, those tasks will register the ID in the Pillow, so that the sleeper hugs the pillow then goes into the hall and make a dream. Got an idea? Cool. So what does the sleeper does in a hall? The answer is to make all kinds of the dream. For example if we want to build the web service to get rest call, and return the JSON feedbacks, we only need to do like the way: Firth, build rest call path in the controller; Second: register the call requirements as a dream; Third, build the sets of the dream in the pillow, Fourth hire a sleeper to hug this pillow, and go to the hall to make a dream process. At last but least: return the dream goods. Any sense? Cool! For this unique instance, you will know that the sleeper was more like a socket, and the hall more like a thread pool, the pillows like the single vision instance, and the sets like a vision storage, the controller and the process those two sections is a common way of the factory model. The steps landscape of the sleeper who makes a dream as bellow figure 1-2.

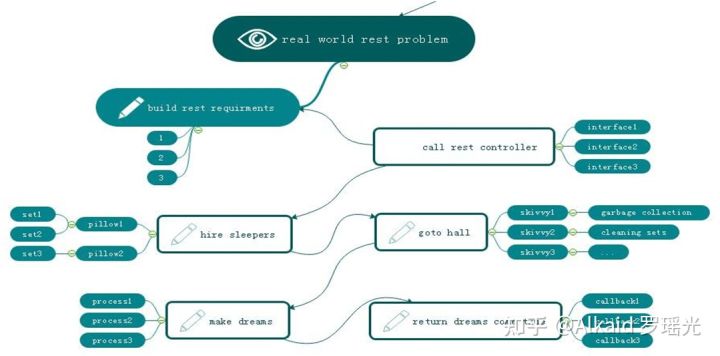


Figure 1-2 VPCS BACK END MODEL

Focus on this landscape, mostly different to the MVC: Model View Controller, MVP: Model View Presenter or other architectures we know before. but is very easy to understand after you read for a while. Too simple. Sleeper makes dreams come true, hall container sleepers, skivvy make up the hall, pillow clear and wake up the sleepers who often lost in finding the way in the dream. Got fun here, but I would hear more argue voice details of my VPCS, desktop App once said: VPCS is good in the concurrent WEB project, but not suitable for the desktop applications. Ok, follow this question, let make a new landscape based on desktop application as below figure 1-3.

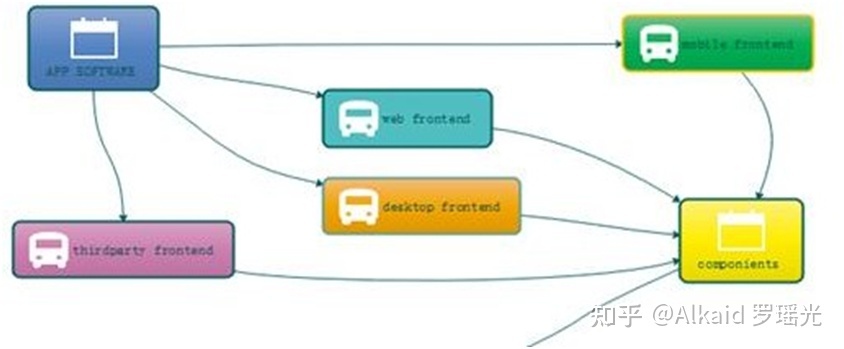


Figure 1-3 VPCS WORK WITH FRONT END

From this picture, we know all of the software can be fast and safe while using VPCS, because it is already separated out the big system into back end and front end two parts. and VPCS keeps safe and fast in the back end section. Compare to the MVC, VPCS will get more cautious and details, and compare to MVP, VPCS also will get more safe and high efficiency. Those factors are why I will make inauguration here. In the common software engineering cycle life times,scientist used to build front end and back end for all kinds of the software applications, because it is easy to control. Why? Frontend only spend time to make design, and Back end for the data operations. Using VPCS system, we don’t care about what they do for the front end, we only fit about what they want. Alignment that gets a blame and fix, then return OK, the restful service developer makes a voice that http functions are concurrent functions. At here, VPCS will say: concurrent functions are safe functions. We guess in the future REST-VPCS will be used in multiple WEB service. Especially in the high speed, efficiency, micro web systems with high level security for example medicine, DNA, cloud server, electronic police system and ecommerce systems etc.

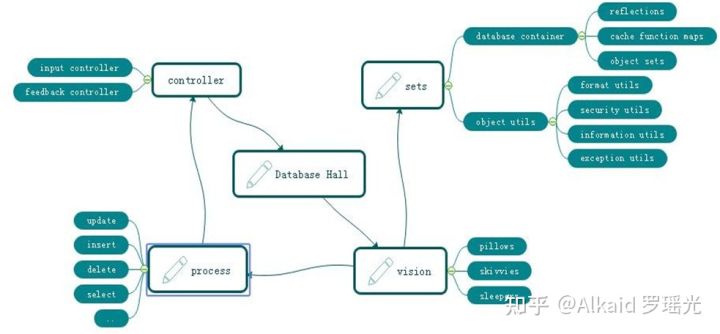


Figure 1-4 VPCS FOR DATABASE SYSTEM

As the figure 1-4, a new method of the Database system designing shows us VPCS is a pretty way for the modern data information management system. Definitely used in the DETA Database system. For this instance, do we get a view that the controller section of the factory model becomes thin yet? Controller only works for the hands transactions, for example that the controller get an input requirement such as select SQL, then immediately call the hall keeper to register this SQL and hire new sleepers to make a result. Because of the VPCS. Once it happened any exceptions, will very easy to awake sleeper and let them get theirs working papers out, finally call skivvy to fork the sets to the fresh sleeper. This method mostly be like a Count Down Latch model, once the sleeper gets the dreams come true, then told the hall keeper for the feedback, hall keeper will makes a type procession to return after everything goes well, This method mostly be like a Cyclic Barrier model.

Questions

How does skivvy doing? please see the figure 1-5 the hall building need a singleton instance like a home keeper but here is a hall keeper, any else, this person is very important for keeping the VPCS safe, because all of the skivvies will be managed by him. You will see, the memory check, JVM garbage collection, disk cleaning, thread status management, deadlock alarm, security protocol all and all in one at here. Mostly like a static class in the VPCS system. If we need to know every thing about skivvy’s work status, ok just call the hall keeper.

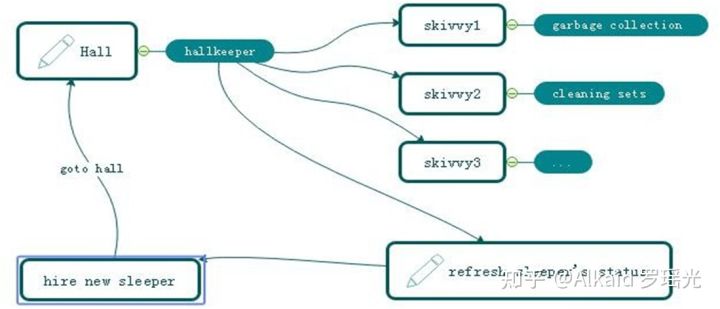


Figure 1-5 VPCS KERNEL

How does sleeper doing? Make a dream? Cool, you shoot!, in the VPCS system, it doesn’t have the definition of the process, everything likes subsets. Immutable or unlined, hall keeper get request from visionary and hire the sleeper, who is likes a thread, get requirement, add those sets in pillow, hugs pillow then go to hall to make a dream, after that then return the callback to hall keeper what they did. Fun yet?What does the sets meaning? Sets, is a format of the data where appearing in the VPCS system. For the static prototype, it used like a concurrent hash table, and list which can be copy base on writing format, the single instance, it always runs in the static function or be liking an interface implementation because need safe at the same time, so that compare to the factory model, it is too simple and without annotation. Everything becomes easy in this environment.The one more question is that so many peoples asked me what does the sequence diagram of the VPCS, because they really want to know why VPCS is faster and safe. Ok, please see the figure 1-6, the answer is absolutely, VPCS main components of the time sequence only contains five aspects. Almost similar like the hotel management. Certainly, we are talking about VPCS software, not for guesthouse. You will see that the rest call only makes the interactions with the hall keeper. And hall keeper got two jobs, one for waiting the fresh sleeper and one more for giving task to skivvy. The sleeper only hugs the relate pillow and make the dreams come true. Fun yet? Cool. VPCS only take cares about how does the sleeper’s imagination and skivvy’s working status. If is the pillow broken? Make new pillow, got lazy sleeper? Get out his working papers, got a cheat skivvy? Fix of fire him, the real source of the java version project for the VPCS only 30kb, we will find more sources or documents from the reference links at the end.

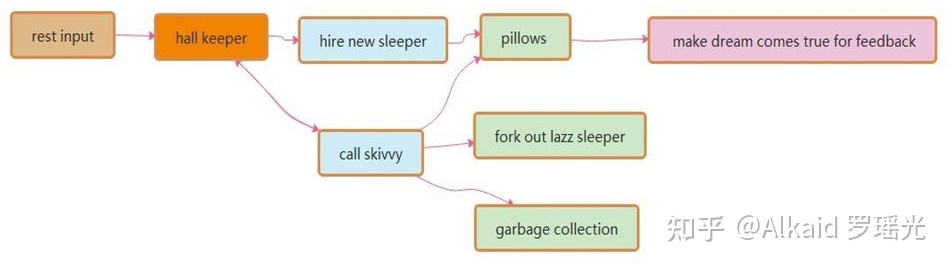


Figure 1-6 VPCS Sequence Diagram

I always be asked by the colleagues that once said: how does the hall build? I answered them, such like the hospital, no one cares about the address of hospital, because they just call the cell phone number when will get a directly feedback. This is why I need a hall keeper role in the gate way. For the instance about figure 1.6.1, this sample is a true demo in the real world for the WEB rest service. Its very important to create a player role such like hall keeper. what would likes about author’s theory? Because of the maintenance. Because of whom, the software build team are very easy to make a maintenance web portal, all of the system current status will be solved on this html page by DEV-OPS.

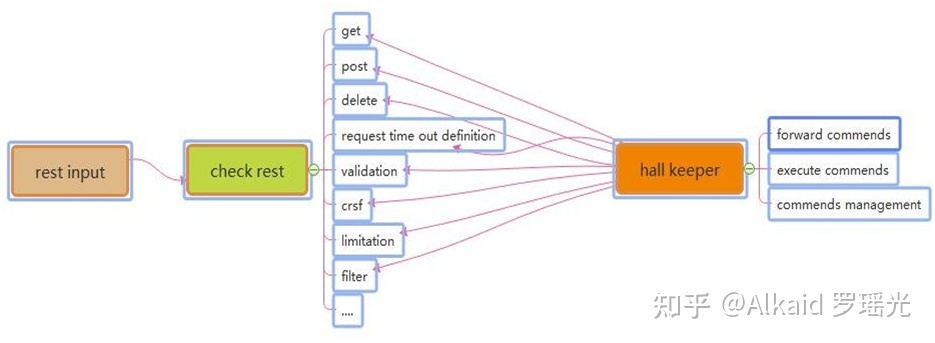


Figure 1-6-1 The Interaction Between Rest Call and Hall Keeper

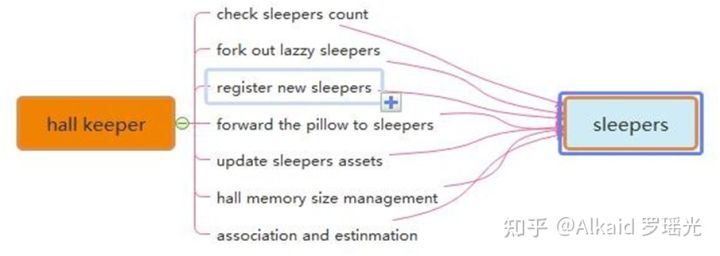


Figure 1-6-2 The Interaction Between Hall Keeper and Sleepers

Many of these software developer also asked me how and why we fork out the LAZZY sleepers excluding their sets. Arthur answered because of the pillows. When the sleepers be hired from the hall keeper, they will get an independently pillows such like static functions. So that sleeper only has their own identify attributes and unique information as the singe instance class. Once they got theirs working paper, the pillows they used will be arranged to the new fresh sleeper, this theory keeps safe, quality and quantity. Like figure 1.6.2.1 VPCS kernel.

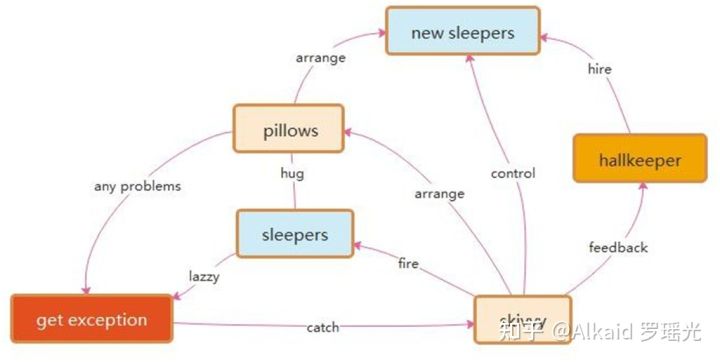


Figure 1-6-2-1 VPCS kernel

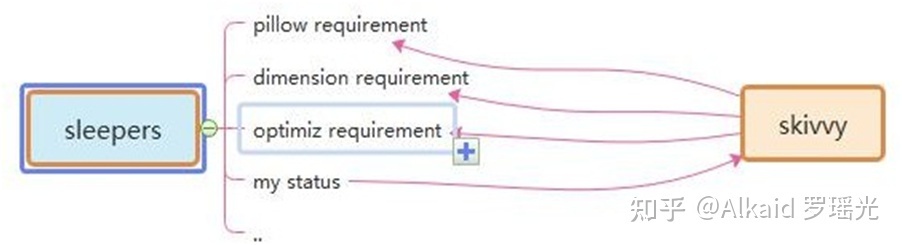


Figure 1-6-3 The Interaction Between Sleepers and skivvy

Many times, I got questions about the DEVOPS, they really worry about VPCS if suitable for their project system maintenance? The answer is absolutely, as the Mr. Ray [274138705@qq.com](mailto:274138705@qq.com) once said: we are DEVOPS, at least we need three important keys in our environment assignments: implementation capacity, transparency and maneuverability. How does the VPCS supports us for daily works? Because of Hall Keeper and skivvy, as figure 1-6-3 and 1-6-4. DEVOPS will get all of these transparency information about project from hall keeper under the encryption and security issues. Also, hall keeper will directly get the rule for DEVOPS by rest calls, then makes to commend to skivvy. All of the information and record logger will be cached by hall keeper, that keeps controllability. The html control page will make an interaction between hall keeper and DEVEOPS, which keeps safe, implementation capacity, transparency and maneuverability. This is my true answer.

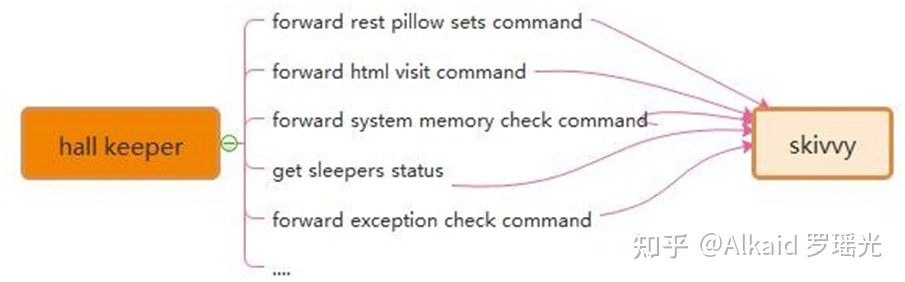


Figure 1-6-4 The Interaction Between Skivvy and Hall Keeper

Recently Mr. Yang [1291244774@qq.com](mailto:1291244774@qq.com) who asked me about VPCS of IOS desktop APP, where and how to avoid the data leakage risks. Because he really worries about the separation between controller and process. Following this topic, my answer that the key is the separation between pillows and sleepers. Due to the pillows all have their own unique ID, skivvy will easily arrange the pillow to new sleeper after the original sleeper who made problems. Make unique ID and arrangement by ID, is the key method. Also for the rest call service, the asymmetrically irreversible combination encryption is one of the best solutions to the data leakage controller. VPCS seems so smart.

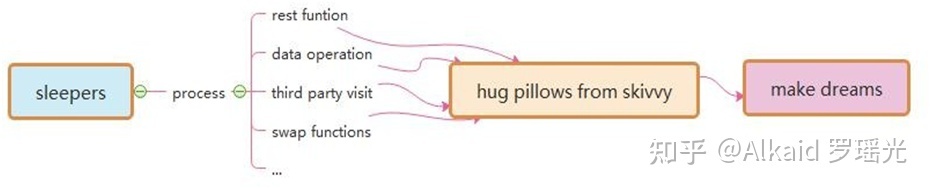


Figure 1-6-5 The Interaction Between Sleepers and Pillows

**3 DETA Catalytic computing**

I got my mind storm for a month in early 2019. How to realize human computing? It's been bothering me for a long time. How to start? I didn't have a clue, so I started to read my notes made in the past 20 years. I got it! Do basic research! According to my notes, I dig some unknown basic knowledge. I am veryhappy, because I have the results of quick sorting by left-right comparison in 2014, the butterfly calculation manuscript of Fast Fourier, the Chinese word segmentation works of Huaruiji, UNICORN ETL, socket stream PLSQL database, etc. and an idea came into being. I think about optimizing them continuously, refining, optimizing and testing repeatedly, and remembering these optimized ideas.

**3.1 DETA Catalytic computing history,**

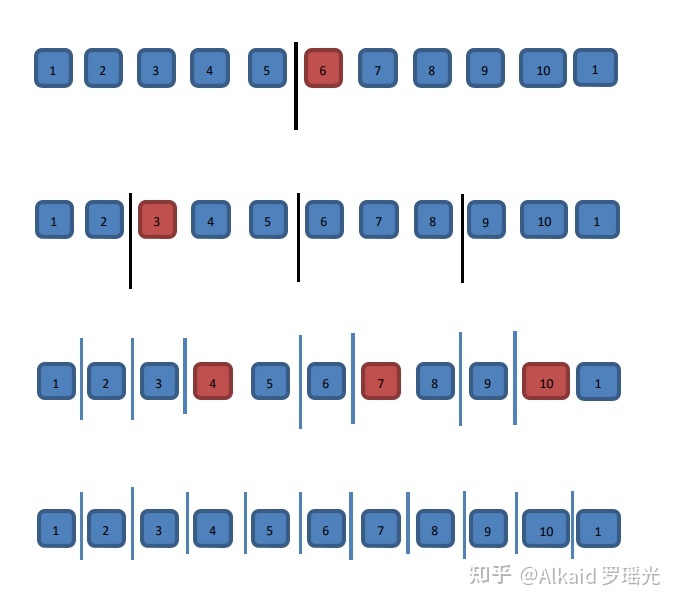
Since of the bright flashes, may I follow the operation method. The refinement method of DETA's first catalytic calculation is first reflected in DETA parser, such as the refinement of semantic part-of-speech analysis, the optimization of flow valve, the irrational conditional transformation of discrete data, the filtering of the same frequency operator, and the filtering of calculation peaks. These optimization methods of human thinking gradually form a system, which not only changes the design mode of DETA's works, but also changes the author's research and development philosophy.

**3.2 DETA Catalytic computing development,**

R&D is not successful every time. In the process of butterfly calculation optimization of Fast Fourier, I coded the features of discrete DCT in complex numbers, which took me one month, but failed. I remembered that I said in Weibo at the beginning that I could speed up the calculation of Fast Fourier by 200 times, but I really didn't give up. Since butterfly calculation optimization was unsuccessful, I tried to sort the small peaks by fast left-right comparison. I was excited when I saw the 10th generation of single machine random double with a sorting speed of 12 million arrays per second of quick sorting. My thought is right, and thinning logic is an important way of human thinking. Here, the author designed an argumentation paper when designing fast word segmentation and extremely fast peak filtering catalytic sorting, as follows:

**Theory on YAOGUANG's Array Split Peak Defect**

Mr. Yaoguang. Luo

稍后继续优化， 罗瑶光

Outline ：In the common software development factory, engineer always did more and more interactions with data structure and math algorithms. Especially in the recursion, convolution, sort and generic loops, scientist likes to find a simple, more sufficiently and alignment way to face the project requirements with the large association. For instance me, I really got a real world problem at this domain while I use quicksort,also for other project such like DETA parser. What is the peak array split defect? How does it count the real world problems? Why need find it and how to get the nice solution? Cool, this paper will cause an implementation about our goals, ok now, keep forward to the context where I talking as below, thanks For more theory DETAils and the source code implementations please check the bottom reference section. Peak, Array, Split, Defect, Recursion, Convolution, Sort, Generic

**Goal one: Quicksort Yaoguang.Luo 4D**

DETAils：

For example the array input as below where we gave 11 digits.

1.1 The first split, we could see the digit-6 will auto arranged to the right part.

1.2 And the second split, we may see the digit-3 will be auto arranged to the right part

1.3 The third split, we may see the digit-4.7.10 will be auto arranged to the right part

2 Thinking:

After the split array showing, we could see clear that the big problem about the asymmetry defect, as I did an annotation of N, so the i of N

will absolutely find a n/POW（2，i）value points, as an insufficiency asymmetry defect model, I fall in thinking…if I do any compute theory as

the same with this model style, for example in the ecursion or inner loops, it will autonomic separate to the 2 different process way, it necessary

to do indifferent flows.

3 Problems:

So, after the above thoughts, I may get any flashes, First, the even and odd digits both are asymmetry while in the Differential loops. For this

noise, I defined as (Tinoise Peak) Second，once we did a split compute under this model, it must get more unfair sets. I defined as (Tinsets defect).

Third, if this model almost in the messy and timer data system, it will catch more time and asserts wastes or exceptions.

4 Solutions:

For the god like, I find three solutions while I currently enrolled in my projects. First: computer logic acceleration, at least it can avoid the

waste of the compute by using inner process optimism. -- To avoid the deep recursion. Second, reduce the compute sets. For any less memory system,

we may reduce more and more memory garbages after we reduce the inner register or temp value sets. Third, we may make an optimization of the function

logic where to instead the old complex functions. Those ways include the condition, algorithm, method or discrete optimization. End, we may use

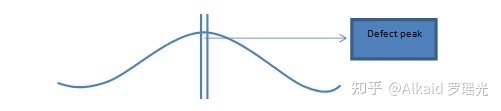
mathematics of double differential, deep definition, acquisition or polynomial to get the solutions.

5 True Instances



贴个最新TopSort5D: [https://github.com/yaoguangluo/sort/blob/master/LYG9DWithDoubleTopSort5D.java](https://link.zhihu.com/?target=https://github.com/yaoguangluo/sort/blob/master/LYG9DWithDoubleTopSort5D.java" \t "https://zhuanlan.zhihu.com/p/_blank)

Let me show the algorithms here,From this code: in a common quick sort way, the recursion based on the average deep split, suppose the initial array length is N{1,2,3…n} is an Odd, so the separate two arrays will cause an asymmetry defect, those timer asymmetry compute peak collection will cause more and more probability problems such like jam, lock, time waste and heap increment. The odd peak binary split as below：How to avoid those timer distinction peaks? I go more absolutely research where focus on these problems, first, differential flows. This flash is not suitable for here, May good for the DETA parser, I will show you later. Second, compute acceleration. Yep, this is a good way, for example find the big X as the code blew, it will cause the while loop ability accelerations.



int x= a[lp]< a[rp]? a[lp]: a[rp];

（int x= a[lp]< =a[rp]? a[lp]: a[rp];）suitable for string swap top sort 5D

Third, De Morgan condition differential as the code below, it will cause the condition ability accelerations.

while(!(a[lp1]>x|| lp1>=rp)) {

At last but the least, value reduce, code optimization both are very important way of the peak avoid filter.

**Goal Two: DETA parser**

6 DETAils：

Last year I help my father to develop the study software about getting the medicine data collection for quick search. I’ m going to try to

build a search engine system, input format is a string, how to get a Chinese string array split?

Convolution: length indicate by marching Nero index tree as below: 2|1|2|3

Convolution: POS indicate as below: n |c |n |adv |adj

Convolution: split

7 Thinks:

While I use this way on my DETA project Chinese separations, I met so many problems, the more and more important problem is the POS frequency peak waste, my POS flow functions will spend a lot of time to do the low prior convolution split condition check first… it cost me a lot of time… after I did a collection of my projects, the results are clear.

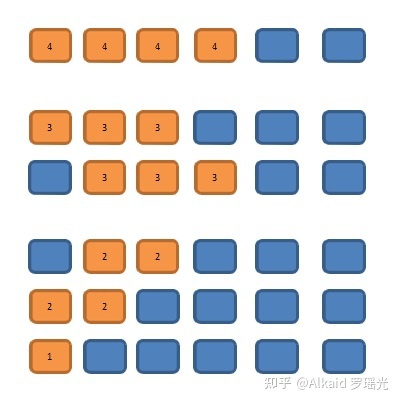
8 Problems:

First, convolution kernel gets asymmetry problems. Second, unnecessary conditions check loops. Third, unimportant heap register values. End, values sets and conditions sets too more.

9 Solutions:

First, format the convolution kernel of the index dictionary tree, for example I use the char ASCII as the index length to reduce the match time of the convolution length indication. Also, I define the POS convolution kernel size less than five. Second, I did a condition frequency statistic, and re arrangement it, at the same time, reduced a lot of the inner sets to avoid the compute pause.

10 True instance:

做成了图片，稍后修改，罗瑶光

11 Finally I developed a convolution String array split way for marching as below: orange color are presplit sets

11.1 check 4 chars slang

11.2 Check 3 chars key word

11.3 Check 2chars normal word

In order to make a compute acceleration, I did 2 string builder array to store a pre order sets.

in order to make a PCA POS acceleration, I did 5 chars marching array to store a post order sets.

It seems the Nero Index, NLP and POS for the PCA separation with convolution kernel marching by using stepwise iterative differentiation got much higher sufficiency.

**3.3DETA Catalytic computing application,**

In order to demonstrate the importance of DETAiled logic, I began to integrate this logic concept into my YangLiaoJing and all my soft works. When I saw 13 million high-accuracy word segmentation per second, 6 million mixed phonetic symbols per second, 12 million double arrays sorting per second, and other amazing works came out, I began to sigh my own cognition. I unreservedly opened up all these ideas and works, which hope aroused humanoid thinking [Resonance Promotions - More Than Promotion - We Care A Lot - Andi Appel](https://link.zhihu.com/?target=http://resonance.at/" \t "https://zhuanlan.zhihu.com/p/_blank) present, the significance of differential catalysis has included seven categories: frequency valve (von Neumann) differential, discrete logic (De Morgan) differential, high-frequency function degradation, conditional refinement differential, executive mode differential, giant system (Qian Xuesen) module differential, and mathematical differential (Newton, Blainez), which is the only way to catalyze humanoid DNA evolutionary algorithm. And made a DETAiled demonstration and summarized as follows

[https://gitee.com/DETAChina/DETAParser/blob/master/wordSegment/org/tinos/engine/pos/imp/POSControllerImp.java](https://link.zhihu.com/?target=https://gitee.com/DETAChina/DETAParser/blob/master/wordSegment/org/tinos/engine/pos/imp/POSControllerImp.java" \t "https://zhuanlan.zhihu.com/p/_blank)

Demonstration of differential algorithm of POS water valve with DETA fast segmentation;

Demonstration topic: Differential catalytic calculation can well observe the execution flow logic of quantum state function. Through statistics, high-frequency function can be advanced gradually according to von Neumann state, and low-frequency logic can be eliminated and screened gradually.

Demonstration result: The demonstration was successful. Reduce the traversal times of irrelevant code. Greatly improve the computing power.

Demonstrating influence: the main way of evolutionary mechanism of humanoid DNA evolutionary algorithm.

[https://gitee.com/DETAChina/DataSwap/blob/dceeb0b06f726d640553964058d85b736354ac89/src/org/DETA/tinos/array/L](https://link.zhihu.com/?target=https://gitee.com/DETAChina/DataSwap/blob/dceeb0b06f726d640553964058d85b736354ac89/src/org/DETA/tinos/array/L" \t "https://zhuanlan.zhihu.com/p/_blank) [YG4DWithDoubleQuickSort4D.java](https://link.zhihu.com/?target=http://yg4dwithdoublequicksort4d.java/" \t "https://zhuanlan.zhihu.com/p/_blank)

Demonstration of The 4th generation example demonstration of filtering sorting algorithm for Luo yaoguang s small peak calculation;

Demonstration topic: Differential catalytic calculation can be combined differentially with discrete mathematics system to filter high-frequency functions where from the digital logic level. Ensure smoothness

Demonstration result: The demonstration was successful. Greatly increase the calculation speed.

Demonstration influence: humanoid DNA evolutionary algorithm can effectively smooth the peak of computation.

[https://gitee.com/DETAChina/Data\_Prediction/blob/master/src/org/tinos/DETA/tsp/YaoguangLuoEulerRingTSP2D.java](https://link.zhihu.com/?target=https://gitee.com/DETAChina/Data_Prediction/blob/master/src/org/tinos/DETA/tsp/YaoguangLuoEulerRingTSP2D.java" \t "https://zhuanlan.zhihu.com/p/_blank)

Demonstration of the second generation of differential TSP algorithm for Luoyaoguang Euler forest business travel ring Demonstration topic:

Differential catalytic computing can optimize the thinking mode of traditional complex logic at cognitive level, and fundamentally change the cognitive process from the beginning.

Demonstration result: The demonstration was successful. Differential catalytic computing can change the traditional cognitive style in some social fields.

Impact of demonstration: Humanoid DNA evolutionary algorithm can effectively select the fastest algorithm module and cognitive module to do calculation in specific fields, and improve computing power.

[https://gitee.com/DETAChina/DataSwap/blob/master/src/org/DETA/tinos/string/LYG4DWithChineseMixStringSort7D.java](https://link.zhihu.com/?target=https://gitee.com/DETAChina/DataSwap/blob/master/src/org/DETA/tinos/string/LYG4DWithChineseMixStringSort7D.java" \t "https://zhuanlan.zhihu.com/p/_blank)

Demonstration of the 7th generation example of Luo Yaoguangs conditional differential sorting algorithm for light image strings;

Argument topic: Differential catalytic computing can unify conditional functions considerably, reduce logic complexity, and continuously optimize and focus.

Demonstration result: The demonstration was successful. Differential catalysis algorithm can split the local modules of the whole function by vpcs logic, and form the purine element of initon operation of DNA peptide chain.

Demonstration influence: the guarantee of autonomous evolution mechanism of humanoid DNA evolutionary algorithm.

[https://gitee.com/DETAChina/DETA\_PLSQL\_DB/blob/master/java/org/lyg/db/plsql/imp/ExecPLSQLImp.java](https://link.zhihu.com/?target=https://gitee.com/DETAChina/DETA_PLSQL_DB/blob/master/java/org/lyg/db/plsql/imp/ExecPLSQLImp.java" \t "https://zhuanlan.zhihu.com/p/_blank)

Demonstration of PLSQL differential compiler for DETA Socket stream programmable database engine;

Demonstration topic: Availability of multi-condition execution of differential catalytic calculation.

Demonstration result: The demonstration was successful. Differential catalysis algorithm can provide reverse observable operation and maintenance guarantee for functional system operation.

Demonstration influence: the comprehensive application practice of conditional differentiation, logical differentiation, high-frequency valve preposition and other functions of humanoid DNA evolutionary algorithm.

These arguments were a year ago. At present, many works have been in a good follow-up state because they are developed as subsystems of the project of YangLiaoJing. Over the years, I have been thinking, what is the final expression of DETA acquisitive logic? I have never stopped exploring, and I have always been absolutely focused.

**4 DETA Finding Initions**

I have been thinking, what is the final expression of DETA Acquisitive Logic? I have never stopped exploring, and I have always been absolutely focused. I must find the final expression of these logic. From 2018 to 2019, I thought that the final expression of logic refinement must not be as simple as AOPM and VPCS. VPCS is just a refinement layer of AOPM, so how can VPCS be refined? So I began to sort out my existing things, my works and soft thoughts. God, I can only make persistent and absolute focus on what I have. I have to make a bet.

**4.1DETA Finding initions history,**

What's under VPCS? What is the essence of a function? At school, I got some basic answers. the primitives of DNA are ACGTU purine and pyrimidine, the primitives of back-end architecture are VPCS, the primitives of thing logic are AOPM, and the primitives of database are IDUC addition, deletion and modification. the primitives of function are IOAON Input, Output, And, Or , Negation, which I can only find in the knowledge structure I can understand and have. how to demonstrate? How to confirm the argument?

**4.2DETA Finding initions development,**

The first demonstration process is DETA word segmentation. In 2019, I continued to refine, optimize and refine the word segmentation, and found an exciting argument. My word segmentation function was continuously split rationally.Finally, a pile of simple combination application fragments of addition, deletion and modification were displayed by IOAON. For the most powerful argument, when I was processing nouns in word segmentation, the final function was formed. Memory takes out 4 words, compares 4 words proverbs, does not? then compare 3 words, does not? then compare 2 words, and does not? then split into single words. This process is summarized in one sentence as a combined decision-making process of adding, deleting, modifying and checking memory data according to John Von Neumann's time flow form. When I think about this, my eyes shine. IDUC is not only the operation mode of database, but also the operation mode of memory data, and it is absolutely focused continuously! ! ! Assuming that IDUC is effective for all data operation modes, assuming it is successful, if it is coded, it is a very strict coding mode of data DNA. I found it! I began to refine my sorting algorithm, word segmentation algorithm, ETL, YangLiaoJing, etc., and found one thing in common. All my works were refined to the rational function level that I could understand, which were small fragments of the combined decision-making process of adding, deleting, modifying and checking linear, multidimensional, database and memory data. These fragments can be coded effectively.

**4.3DETA Finding initions application,**

With this in mind, I have determined a ternary mapping coding mode of DNA to ETL neuron nodes, AOPM -VPCS- IDUC 3D coding mode. 4\*4\*4 Then each primitive is a 64-bit space, which is the computing primitive I have been looking for decades.

**5 DETA DNA decoding**

If the AOPM -VPCS- IDUC 3D computational neuron mapped by DNA IDUC is established, how to decode it? I thought about what I have, about YangLiaoJing! It is the only way that I can do at present to construct the system of YangLiaoJing and demonstrate this idea and technology. It is still the absolute focus of that sentence.

**5.1DETA DNA decoding history,**

At present, what we know is that DNA pep-tide group has billions of long, double chains and 24 pairs of chromosomes. there are five primitives of ACGTU. if ACGT can encode human higher intelligence logic, then human-OID data DNA with IDUC unit can also write hundreds of thousands of business transaction processing logic of AOPM VPCS. these two logics do not conflict.

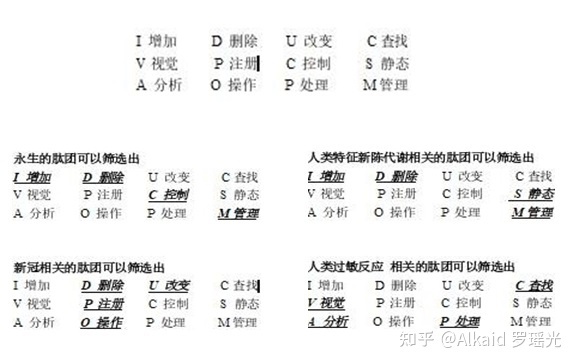
**5.2DETA DNA decoding development,**

These two logics do not conflict. Are they one? I got an exciting argument, and I had to find an argument, so I screened my study and work style in recent 20 years, my own thinking style and the execution logic style of my software, hoping to find a negative theory to overturn it, but unfortunately I couldn't find it. So I followed up and re-examined my soft works, optimized them, and found that once optimized to the edge of rational function to irrational function, they were all linear. Small fragments of the combined decision-making process of adding, deleting, modifying and checking multidimensional, database and memory data. These fragments can be coded effectively. AOPM -VPCS- IDUC seems to explain all the answers I want.

**5.3DETA DNA decoding application,**

In order to overthrow my argument, I began to look for arguments everywhere to attack this argument. First, I found the topic of eternal life. According to AOPM -VPCS- IDUC, IDMC is true. Since it can be perfectly explained, I found the topic of infection in COVID-19, that is, DUOP is true, which is an exciting conclusion! I have been searching for answers to all the problems for decades from AOPM -VPCS- IDUC, so I started mapping and coding as follows:I got a clear

**DNA theorem:The essence of DNA is a combination indexing link list of four meta-operations of adding, deleting modifying and Querying data. 10-04-2020 DC**



**6 IDUC DNA and Its Applications,**

These are all the later stories. The application is too wide. First of all, my ETL began to expand in the three-dimensional direction to better serve medicine. Secondly, virus immunology and immortal virus exploration will never stop. Why ETL is used as the expansion point is inspired by my OSGI paper on October 17, 2013. It is as follows

**The Darwin's Theory of The Artificial Intelligence**

In the latest knowledge engineering structure, the traditional expert system occupies a dominant position, but the world's demand system is in a changeable operating environment, so the data persistence theory is a goal to strive for. Artificial intelligence software, too, can't escape the disadvantages brought by natural updating. Where artificial intelligence will go, it will be planned naturally. Just like Darwin's theory of biological evolution, the new intelligent system standards are naturally selected by needs, which is the central idea I want to express. In the past 50 years, some classic software can't escape the choice of demand, and finally it turns yellow and dim. Of course, some enterprises rewrite and upgrade their products desperately. Because of the aging of core developers, new reformers can't master the original development ideas and theories. Finally, the quality of products suffers a huge impact and suffers heavy losses. A new software development theory needs to be confirmed, which is my thought. Software, too,needs an evolutionary system with self-artificial selection.

Through the recent construction, design and coding test of UNICORN AI software, I found that many computer theories created by fantasy have great differences in actual programming analysis. I used JAVA-based language, and I found that the inheritance of JAVA did not meet the language standard with evolutionary thought, but its methodology in this initial evolutionary standard test was far superior to C/C++. I didn't bring any troubles to my actual programming when I wrote JAVA program in C style, but JAVA still needs to be improved. For example, you abstract a parent class, and the variable function of your subclass still needs to be written in the way of "OBJECT parent class = (subclass) parent class" to make subclass operation. If grand children inherit sub classes, how can OBJECT get grand children? (I use the OBJECT subclass to inherit the parent class, and then the OBJECT subclass = (grandson) subclass. In this way, the grand children get the operation), but this is a big problem of dynamic memory structure allocation!

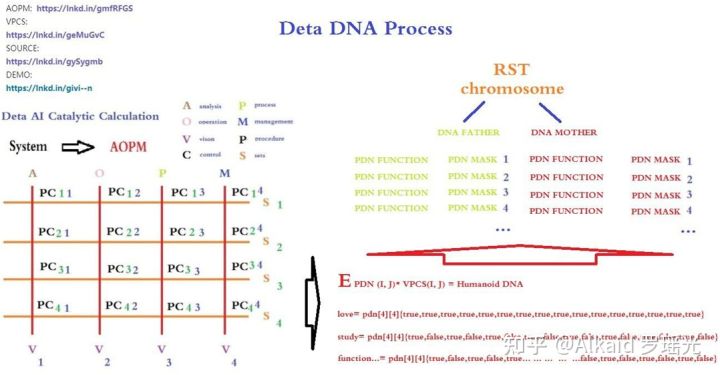
The design is rather cumbersome. JAVA still stays at the level of primary language evolution, and does not have advanced evolutionary ideas. Secondly,if a subclass has more than one grand child, only the subclass can operate, and the parent class cannot operate accordingly. This is also a criticism. Is it realized by adding OBJECT subclass = (grandson) subclass and OBJECT parent class = (subclass) parent class? This is even more complicated.

Through the above description, I have my own views, but I still chose JAVA, even though it is cumbersome, but there is no mistake, because it will be more cumbersome to implement in the underlying language. There are more traps. It is a natural choice for artificial intelligence to choose JAVA. JAVA and C# are both high-level languages, but JAVA's personality is born to deal with data, because JAVA was a WEB language in its early days, and WEB has unique advantages in dealing with data information, which is a real example of JAVA evolving into a data analysis language. C# has been improving itself in this problem, just like JAVA, even like JAVA, but there is no system to evaluate it. The WEB data engineers who applied JAVA in the early days will not transfer to C#, so the biggest advantage of C# is that it is only applied to controls on WINDOWS.

Through this description, it only proves that the greatest advantage of any language is only reflected in its creative theory and thought at the beginning of its birth. Therefore, JAVA and C# are not comparable at all. Because their original creative theory, system and ideological structure are different. If JAVA and C# really fail, finally, through the prediction of evolutionary thought, JAVA will go in the direction of graphics, big data analysis, WEB and C# should go in the direction of interface, control and WINDOWS device integration. The evolution of artificial intelligence software is mainly divided into update of parent class, variation and inheritance of subclass.

JAVA is perfect for dealing with subclass functions, and people who have used JAVA to develop large projects are quite experienced in dealing with interfaces and inheritance. But is there any variation in JAVA? It can be said that there is no, for example, when the parent class PUBLIC attribute 1 = 0; Sub classes can't have the PUBLIC attribute 1=1, which is a mutation failure problem. JAVA is flexible, but not as flexible as scripting language. Secondly, I want to say that the variation of JAVA is a variation with quotation marks. Its characteristic is that sub classes modify the functions of the parent class, and sub classes of JAVA can modify the processing procedure of functions with the same name of the parent class. However, you have to make the subclass and the parent class have the same function names. This is a JAVA default mechanism, which executes the same name of the parent class first, and then executes the same name of the subclass. Then return to the parent class, and then return to the procedure of. Therefore, the function with the same name can be modified in sub classes, thus ensuring parameter variation. In this way, the software is also very flexible and unique in the actual writing process.

Finally, I have a deep experience through the expression of language evolution thought and program evolution thought mentioned above. Every language needs its needs if it is to be deeply rooted,and its functions should be selectively evolved in the needs. Otherwise, this is the biggest reason why languages have been eliminated. I don't like to see various languages emerge one after another in today's world. This is the biggest criticism that many languages have not evolved and can't reflect their needs. Secondly, languages need to be extended, and the appearance of API class libraries and some architecture systems of high-level languages is a good proof of extension. Finally, variation is similar to scripting language, which is flexible and convenient.What about software? The same is true for software. It is particularly important to choose a language that suits your own needs. Secondly, the architecture of the software should have loose coupling, which is similar to OSGI and FELIX. The OSGI idea of KNIME is consistent with that of LIFERAY. Although the API design style is different, the effect is very thick. Biology needs Darwin's thought, and artificial intelligence also exists, which is the basis of demand persistence. This is also the basic condition for my research and development of UNICORN AI platform. Now I have enough confidence to continue to focus on the argument of making ETL mapped by my DNA code with evolutionary system reuse the perfect guarantee requirement persistence. On how to use ETL to map the code, I will go back to the previous year again and analyze the design idea of this picture at that time as follows



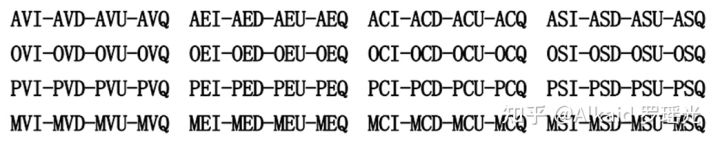
Yes, I have an idea already! ! Exciting, ETL node three-dimensional classification. this vocabulary

**7 IDUC VPCS AOPM 3D Nero Cell and Its Applications**

These are the following words. ETL begins to expand in the direction of 3D. First of all, I want to design the 3 D functional area of neurons based on the DNA mapping of DETA IDUC. This is the real human-OID independent thinking way that I can understand.This is the first index application idea of DNA coding manual in human history.

I will Change Org.lyg.node.medcine.addchufangattributeH.jar into Org.node.a.v.c.u.medcine.addchufangattributeH.jar

This a.v.c.u will form a DNA mapping system code for analyzing visual control changes, which is convenient for future evolutionary optimization tests. After that, I will systematically encode these ETL index mapping sets into DNA index chains for YangLiaoJing. The ideas given to me in this paper are all creative ideas. Thank you for everything. I can first design AOPM VPCS IDUC INITONS 64-bit single chain for the integrity of coding, such as



This index mode, even though it is not the final index of organic DNA of human beings, has become the first mapping execution mode of human-OID artificial design representing evolutionary encoded DNA.

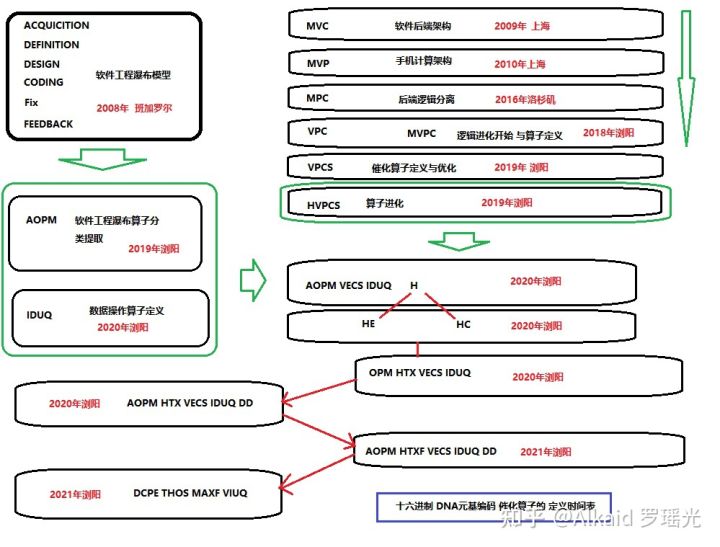
**Not The End:**

In this paper, I spent 20 years of basic study, 7 years of work practice, 2 years of open source implementation, 18 corresponding data sub-projects, 6 data fields, 2 big data works, 7 artificial intelligence papers, and gradually demonstrated some essence, such as The essence of DNA is a combination indexing link list of four meta-operations of adding, deleting, modifying and Querying data.Execution mode of neuron calculation: a neuron time series calculation chain of specifical function calculation by mapping of DNA coding index reflection.

The essence of adapting to the environment is that DNA coding indexes map related neurons compiler link to achieve better addition, deletion,modification and query the environment data.

The essence of human-OID evolution offspring: the offspring produced by optimizing the hybridization of the same coding logic part in the DNA encoding index chain mapped and retained by the neuron calculation method of data efficient processing.

It seems that the topic will never end, so I might as well boldly put forward new arguments, continuously and tenaciously focus and demonstrate, and I still enjoy it.

目前，映射算子最终成为了元基算子。罗瑶光

**8 Refer** (原文文章太长，著作权文件中含有完整refer清单)

第八章\_肽展公式推导与元基编码进化计算以及它的应用发现.

initon定义，肽元基单位如AOPM VECS IDUQ TXH DD（**F 元基不在此章出现**）

**TVM定义, 离散语义肽虚拟机**

**PDW定义, 离散语义肽元基词汇**

**PDC定义, 离散语义肽元基编码**

**PDN定义, 离散语义肽元基结构**

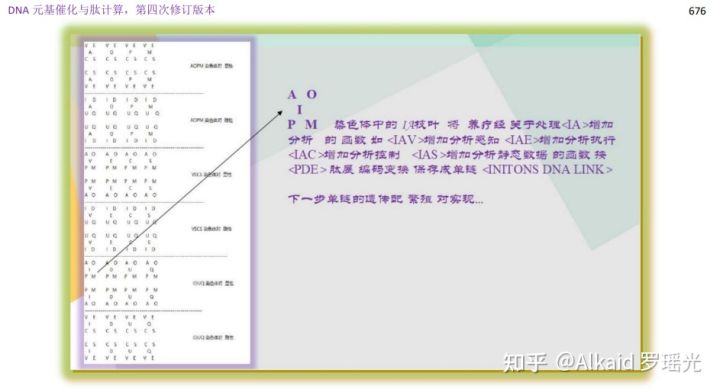
**PDE定义, 离散语义肽展公式变换**

**PDS定义，离散语义肽展逐级丝化变换**

1 DETA INITONS classify/德塔元基分类。refer page 674

2 DETA INITONS PDN words root/德塔元基分类词根。refer page 676

2.1 元基染色体观测。refer page 676



3 DETA INITONS PDN words/德塔元基分类词典。refer page 677

3.1 元基染色体分类的编码根。refer page 677

4 DETA TVM/德塔词典肽翻译虚拟机。refer page 678

4.1 元基染色体的分层级表达。refer page 678

5 DETA TVM applications/德塔肽翻译虚拟机应用技术。refer page 679

6 DETA TVM PDC/虚拟机应用优化。refer page 680

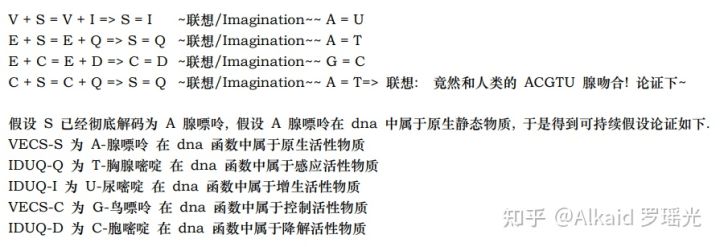
6.1 元基文本加密方式观测。refer page 680

7 DETA TVM PDE/德塔肽翻译推导。refer page 681



7.1 元基进行离散数学推导观测（狄摩根定理）。refer page 681

7.2 ACGTU解码。refer page 682

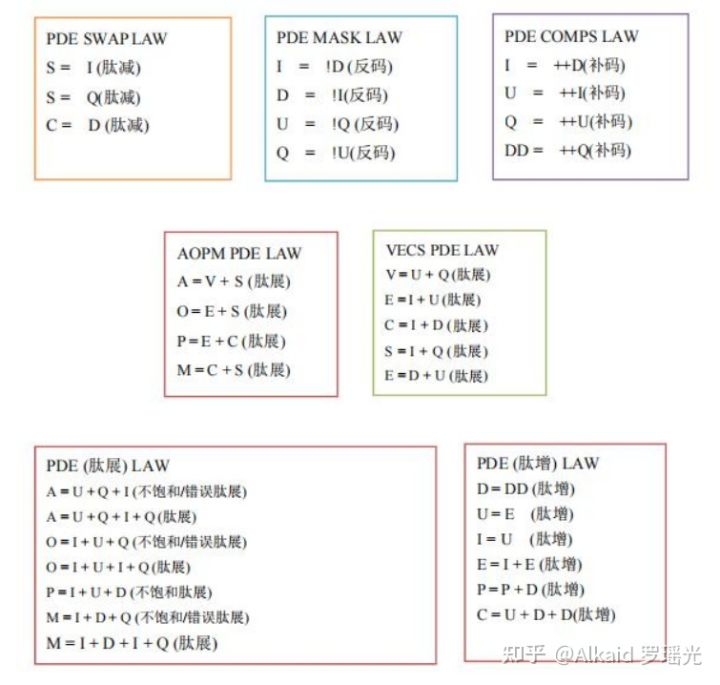


8 DETA TVM PDC functions/德塔肽推导函数化。refer page 683

8.1 三元PDC进行意识词汇的设计。refer page 683

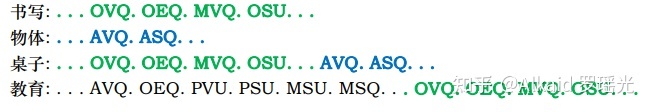
9 DETA TVM PDC function optimization and PDE/德塔肽推导函数逻辑优化。refer page 683

9.1 肽展公式推导集 。refer page 685

肽展公式PDE

10 DETA TVM PDE Logic/德塔肽推导函数逻辑优化成肽展公式化。refer page 686

11 DETA TVM PDE and its application/德塔肽展公式应用论证技术。refer page 687



12 TVM humanoid life Research/应用在类人生命进化中。refer page 687

13 Eternal Research/应用在类人生命永生探索领域。refer page 691

**章节的著作权文件列表：**

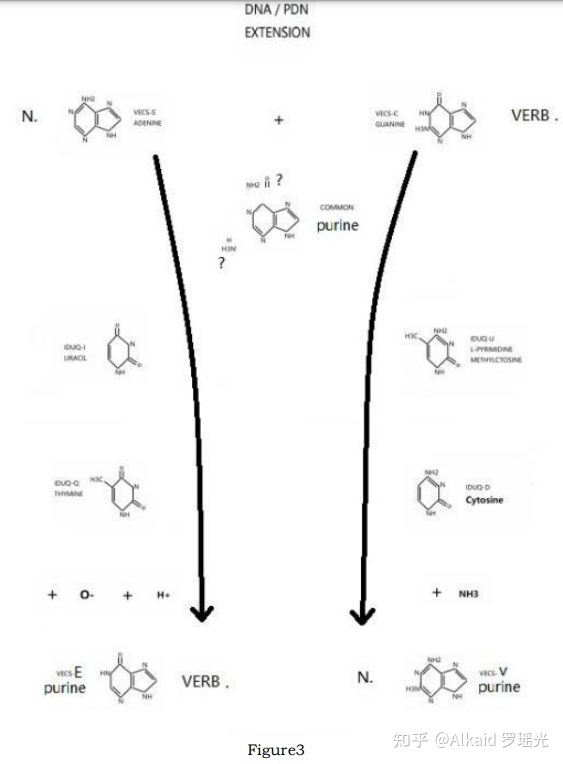
第九章\_DNA催化与肽展计算和AOPM-TXH-VECS-IDUQ元基解码

语义逐级表达的方式，十五元基解码。

1. 推导与定义:甲基胞嘧啶在 DNA 编码和肽计算中具体定义为 IDUQ-U 变嘧啶。refer page 695

2.推导与定义:2 氨基腺嘌呤在 DNA 编码和肽计算中具体定义为 VECS-V 变感腺嘌呤。refer page 698

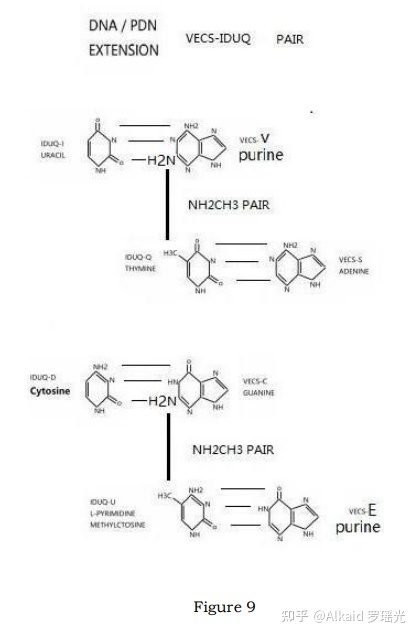
2.1 PDN extension 腐蚀。refer page 699



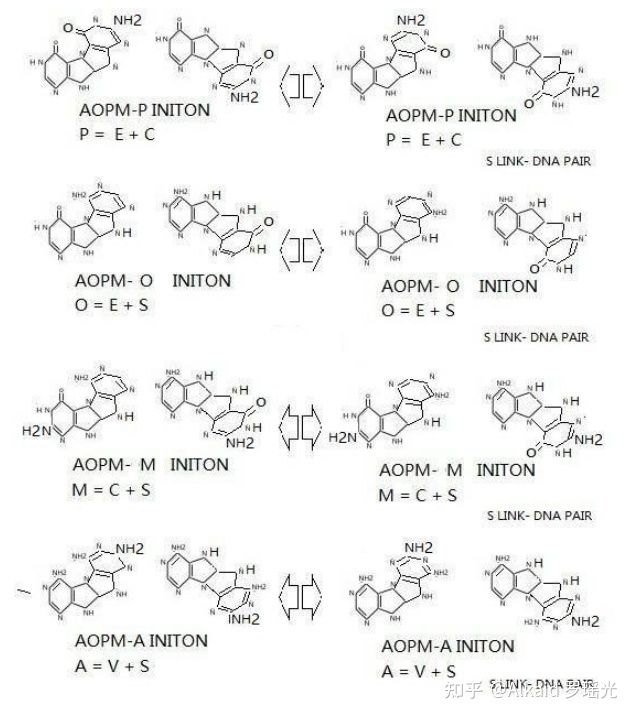
3.推导与定义:次黄嘌呤在 DNA 编码和肽计算中具体定义为 VECS-E 尿变嘌呤。refer page 699

4.推导与定义:AOPM-A 变胸腺苷, AOPM-O 尿胞变腺苷, AOPM-P 尿胞变鸟苷, AOPM-M鸟腺苷的S形螺旋纹血氧峰触发器分子式催化计算严谨完整过程。refer page 700

5. 推导与定义:VECS-VECS 嘌呤对, VECS 嘌呤弧, VECS-IDUQ 碱基对, IDUQ-IDUQ嘧啶对的催化模型。refer page 701



5.1 上下 旋弧 rotation。refer page 701

AOPM rotation

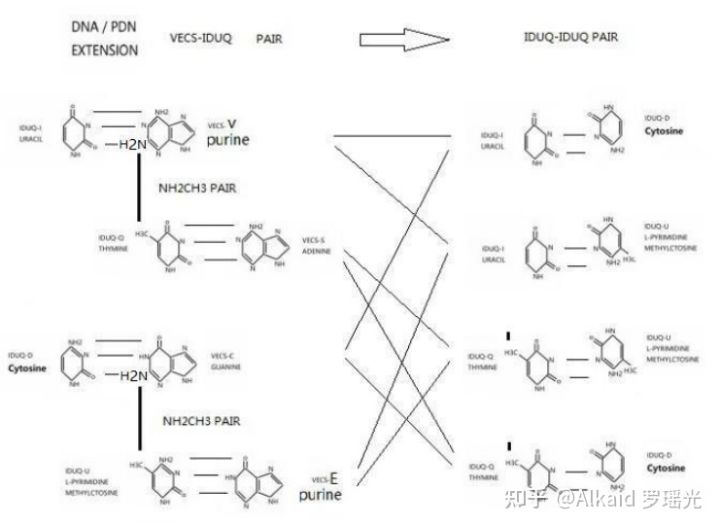
5.2 S link dna pair。refer page 704

6.推导与定义:次黄嘌呤, 尿变嘌呤 VECS-E=IDUQ-U 变嘧啶, 甲基胞嘧啶 E=U 全新DNA计算碱基对。refer page 705

7.推导与定义:2 氨基腺嘌呤, 变感腺嘌呤 。refer page 705

7.1 VECS-IDUQ pair。refer page 705

7.2 VECS-IDUQ pair rotation。refer page 706

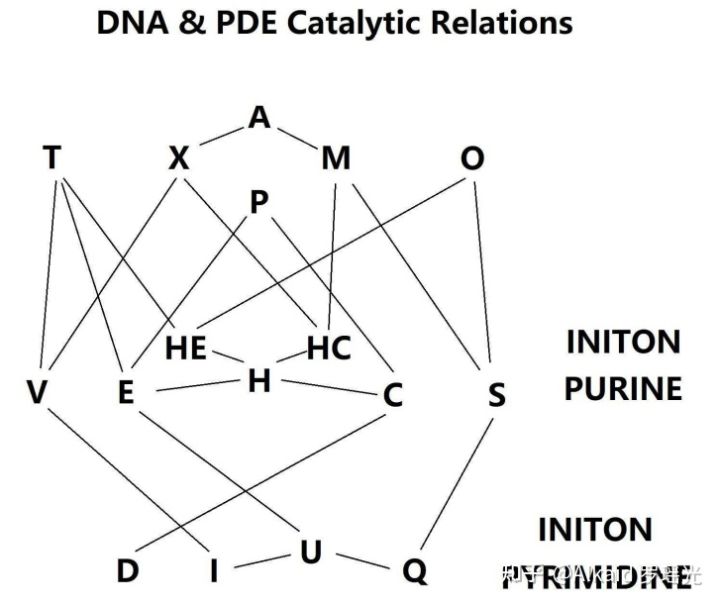


8.推导与定义:碱基对 Rotation 观测与黄嘌呤在 DNA 编码和肽计算中具体定义为VECS-EC尿变鸟嘌呤。refer page 706

8.1 VECS-VECS pair rotation。refer page 707

9.推导与定义:尿变鸟嘌呤, 黄嘌呤肽展计算 AOPM-OP-T 变感腺尿变苷与 AOPM-OP-X 变感腺鸟苷。refer page 708

9.1 DNA and PDE catalytic relations。refer page 708

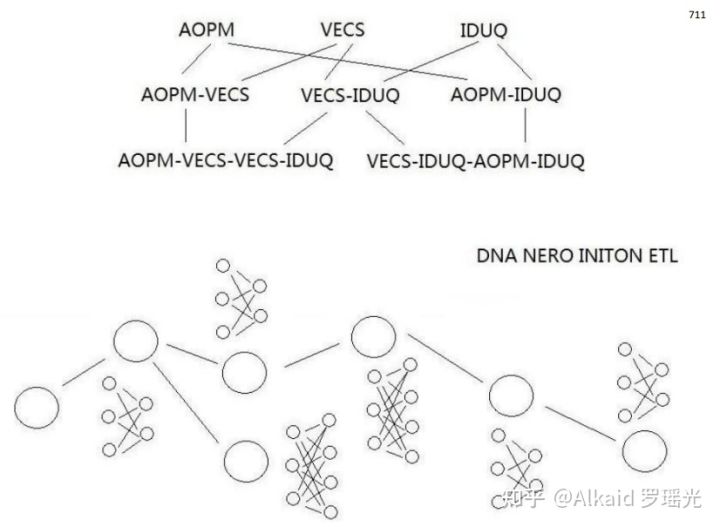


10.归纳与定义:DNA 与 TX-H-U 元基解码。refer page 709



11. 推导与定义:DNA 元基催化计算与 ETL 肽展神经网络计算流。refer page 710

11.1 rotation S link encoding & decoding。refer page 710

PDE ETL

**章节的著作权文件列表：**

第十章\_DNA非卷积视觉技术

定义：**非卷积腐蚀视觉，一般指在不采用卷积内核进行图片低速计算的方式下，图片每个像素仅仅做一次遍历来模拟酸碱腐蚀肽展公式计算，目的是 达到人肉眼观测不到而又难以辨认的色阶群进行观察拉伸。用于极速图片观测识别领域。**

定义人 罗瑶光

测试原图来自医学教材

**DNA非卷积视觉技术原理**

1 DNA非卷积视觉技术将图片像素0~255的区间捕获后，进行元基进制变换，产生了离散色阶。

2 这个色阶，与之前像素亮度色阶完全不对称，利用这一点不对称，可以将邻近像素差不大的像素团进行颜色差 的拉伸。

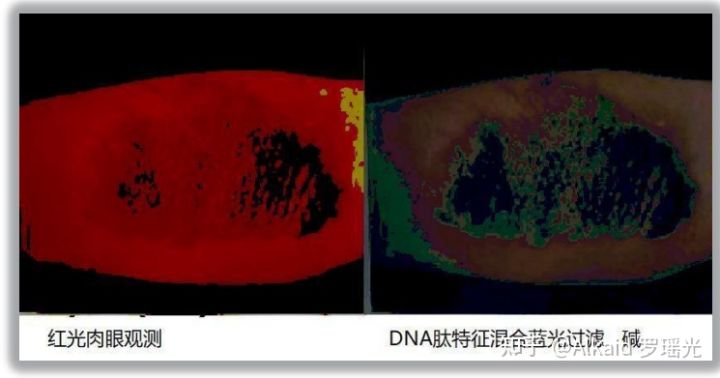
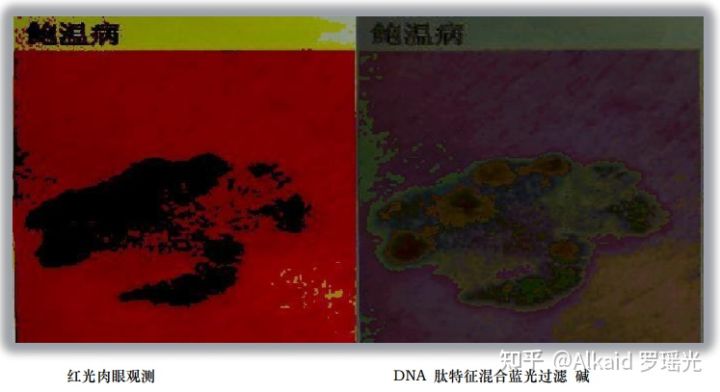
3 颜色差 的拉伸方式 为 有系统的 10进制色 变换为 元基色 进行固定的 酸碱 更换。实践观测后发现价值巨大。

**DNA非卷积视觉技术原理**

1 DNA非卷积视觉技术将图片像素0~255的区间捕获后，进行元基进制变换，产生了离散色阶，

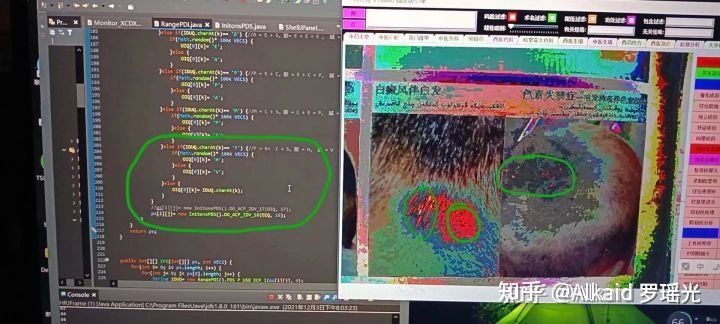
2 这个色阶，与之前像素亮度色阶完全不对称，利用这一点不对称，可以将邻近像素差不大的像素团进行颜色差 的拉伸。

3 颜色差 的拉伸方式 为 有系统的 10进制色 变换为 元基色 进行固定的 酸碱 更换。实践后发现价值巨大。

测试原图来自医学教材测试原图来自医学教材

**肽腐蚀，**

1 DNA非卷积视觉 用元基的酸碱变化规律定义为肽腐蚀。refer page 723

测试原图来自医学教材

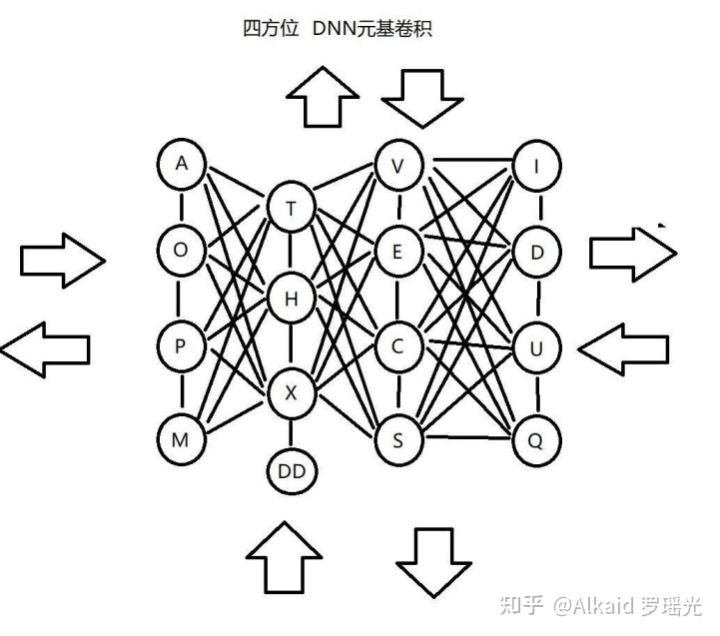
2 DNA非卷积视觉的肽腐蚀 需要将10进制数字变换成生化进制数值。refer page 724

3 DNA非卷积视觉的肽腐蚀观测体征体现在颜色区间上。refer page 735 ，742

4 DNA非卷积视觉的肽腐蚀浓度用概率百分比来标识。refer page 756 monitor.fac x y..

**元基的视觉叠加与表达方式，**

1 视觉流计算。refer page 756



2 animation动画。refer page 744

3 颜色的腐蚀精度调节。refer page 757

4 肽展公式的应用。refer page 723

**时序视觉模拟机，**

1 线性神经网络卷积计算 。refer page 772

2 卷积计算的方式。refer page 773

3 非卷积视觉的应用。refer page 756

图片识别应用，测试原图来自医学教材

**费洛蒙的计算方式，**

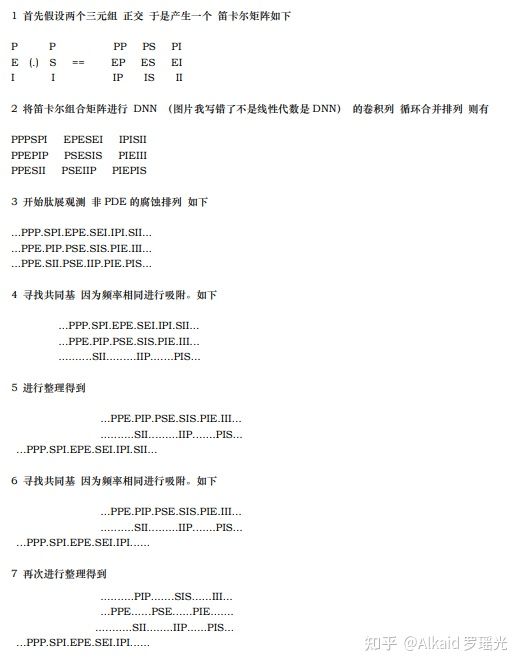
1 CNN卷积元基PDC扩展。refer page 774

2 邻近元基PDC代谢 共同基。refer page 774

3 PDC链结构rotation。refer page 774

4 丝化散开与腐蚀。refer page 775

（**下图的丝化是元基的概率组合的归纳如矩阵的新陈代谢模拟发散。不是肽展公式PDS丝化过程。下图的1和2，作者认为是一种比较合乎情理的又具有代表性的概率矩阵组合归纳。罗瑶光补充20220307）**

罗瑶光的费洛蒙计算发散

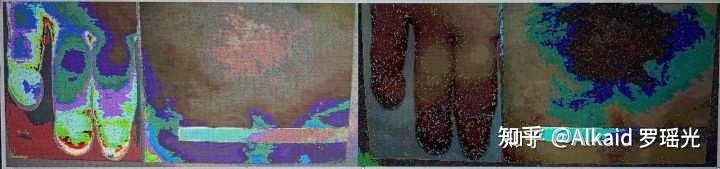
**应用**

1 舌诊观测应用。refer page 736

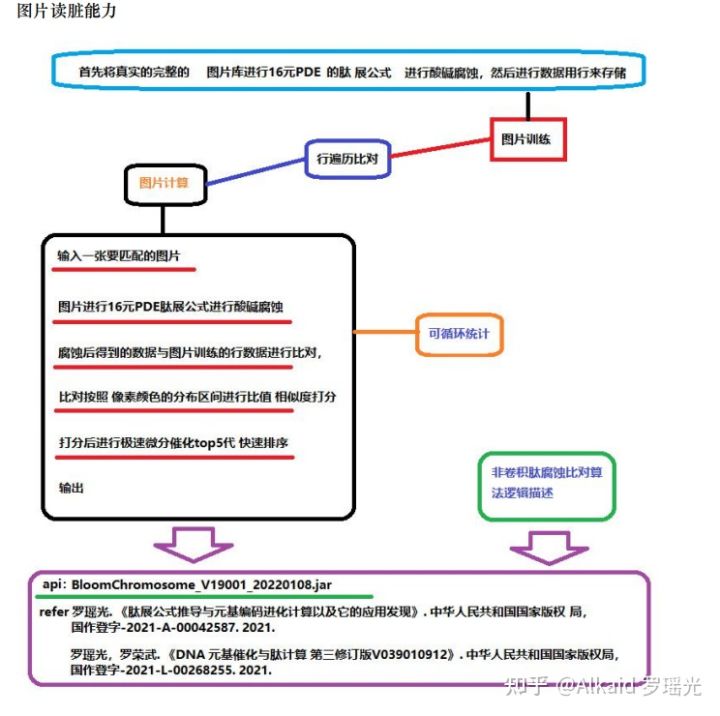
2 骨CT观测应用。refer page 735

测试原图来自医学教材

3 皮肤病观测应用。refer page 下册156，下册157

测试原图来自医学教材

4 图片读脏 应用逻辑



**章节的著作权文件列表：**

第十一章\_DNA\_ETL与元基索引ETL中文脚本编译机.

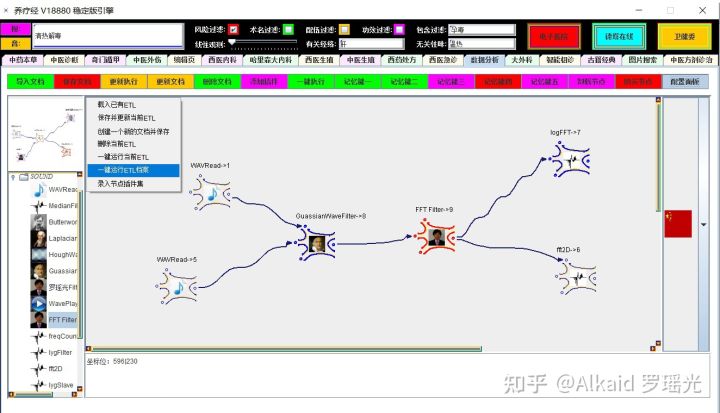
**[养疗经 DNA 元基 版本tin shell 集成 测试 成功](https://www.zhihu.com/zvideo/1426319837153411072" \t "https://zhuanlan.zhihu.com/p/_blank)**

[](https://www.zhihu.com/zvideo/1426319837153411072" \t "https://zhuanlan.zhihu.com/p/_blank)

[Alkaid 罗瑶光的视频](https://www.zhihu.com/zvideo/1426319837153411072" \t "https://zhuanlan.zhihu.com/p/_blank)

[· 14 播放](https://www.zhihu.com/zvideo/1426319837153411072" \t "https://zhuanlan.zhihu.com/p/_blank)

**ETL元基编码方式,**



1 DNA\_ETL的编码继承了德塔数据库的语言编译机。refer page 413,788

2 DNA\_ETL的编码字符串可以自由设计，如中文描述。refer page 834,835

3 DNA\_ETL的编码行可以集成在节点中 etl单个 执行。refer page 782

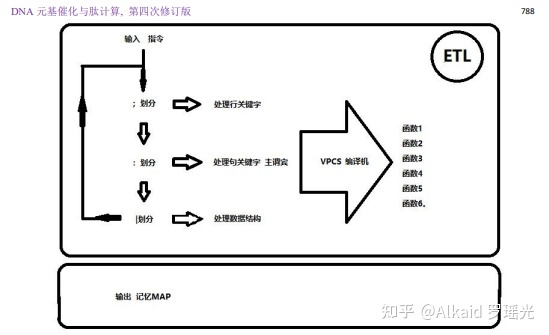
4 DNA\_ETL的编码可以拆卸成节点模式单行进行 etl流 执行。refer page 784

**PLETL语言，**

1 PLETL语言 继承了德塔数据库的语言编译机语言。refer page 377,786

2 PLETL语言 扩展了德塔数据库的语言编译机语言，如TCP, REGEX 应用等。refer page 784

3 PLETL语言 支持多语种 命令设计。refer page 789,790



4 PLETL语言 节点流编译机 可模拟神经网络语言 做计算需求。refer page 783

**Tinshell，**



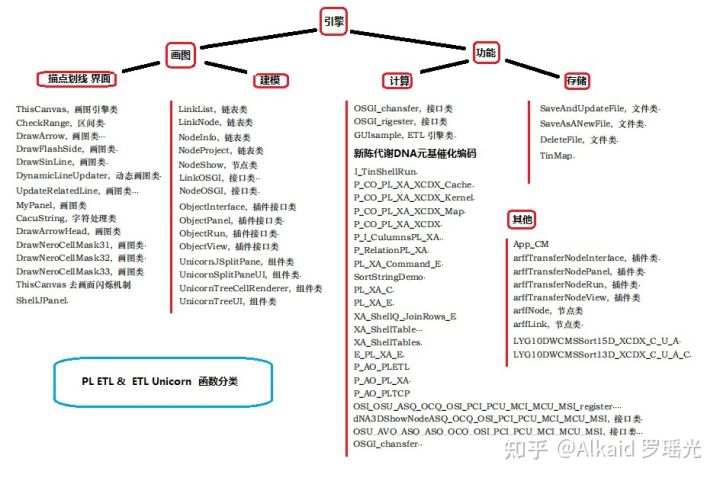
1 Tinshell是封装在 PLETL语言下的基础组件。refer page 860,877

2 基础组件体现在 脚本的编译和执行。refer page 786

3 Tinshell 采用 德塔数据库的语言编译机 进行改装。refer page 788~835

4 Tinshell 主要用于脚本语言的输入和 计算输出 的 IO计算。refer page 782

**编译机的进化，**



1 德塔编译机，最早取自 德塔socket流可编程数据库系统的 plsql编译机。refer page 377

2 德塔编译机在设计tinshell的时候从数据库中分出来做脚本编码编译机。refer page 786

3 德塔编译机在脚本编码中开始扩展，如和etl结合，和tcp结合等。refer page 783

4 德塔编译机在肽化索引后，将用于神经元 etl节点网络计算中枢模拟。refer page 783,784

**osgi插件的肽化方式，**



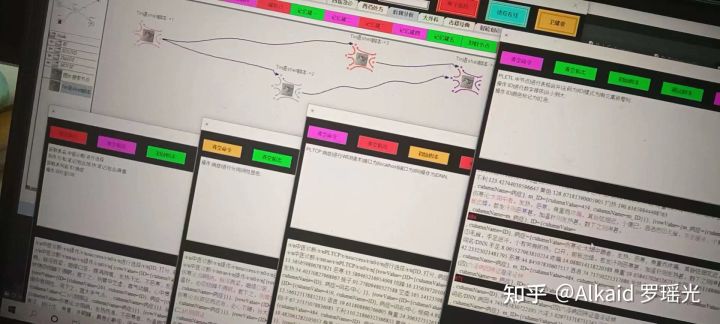
1 osgi插件最早作者设计为了 将节点进行像knime那样 导入。但一直没有实现，只是继承而已。refer page

2 作者2017年5月 在洛杉矶与 印度的 印佛西思 2个月的电话+citrix经理面试时候，被频繁询问classloader技术。于是笔记。refer page 291 Class<?> myclass = loader.loadClass, 作者的最早classloader思维不是来自cnblogs chinaxin。

3 作者2019年开始尝试并真正的 classloader jar实现 节点插件化。refer page 781

4 最近开始肽化索引，用于 classloader识别标记与节点文件分类。refer page 781

**神经元计算模拟 应用**



1 DNA\_ETL的 神经元计算 是一种有向 节点拓扑计算。 refer page 786

2 DNA\_ETL的神经元计算中节点是一个载体单位，不再是计算单位。 refer page 782

3 DNA\_ETL的计算单位是单一一句tinshell 命令。 refer page 783

4 DNA\_ETL的tinshell命令可一句 或者 多句 载入 一个 和 多个节点中。 refer page 784

**章节的著作权文件列表：**

第十二章\_DNA语料数据库加密技术.

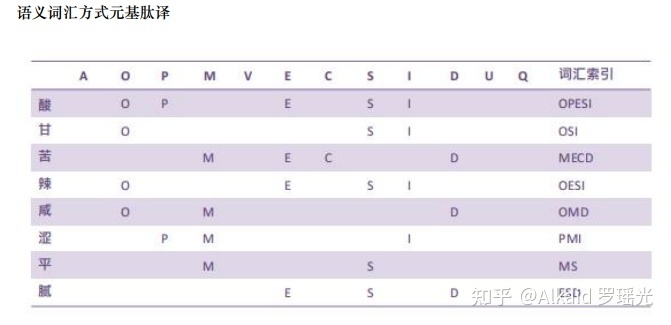
**DNA 元基加密，**

1 DNA 元基加密 包含物理元基加密和 非物理 的 语义元基加密.refer page 900,1015

2 物理元基加密，可理解为将元基编译成密码子，通过算法将密码子替换原文。refer page 900

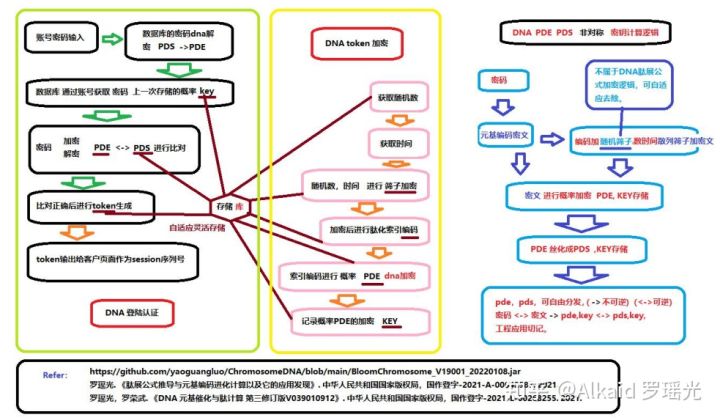


3 语义元基加密，可理解为将文字进行语义肽展公式变换，然后通过酸碱概率二次加密.refer page 907,910



4 酸碱概率的几率成为概率钥匙，用于密文的解密和对比检测。refer page 915~

**非对称概率钥匙加密，**



1 非对称概率钥匙加密 利用的是 肽展公式通过酸碱概率二次和三次加密，加密次序的拓扑过程产生的不稳定性编码如 1变2, 2变1，产生不可还原的因素。refer page 936 A=V+S 1变2,

2 于是这个加密的概率钥匙可以进行前序解密， 但不可后序解密，具备了非对称加密的条件。refer page 1016,1017

3 非对称概率钥匙加密适用于各种不可逆的加密场景中，如互联网登陆认证。refer page 1017

4 非对称概率钥匙加密可以物理与语义进行组合加密，增加安全级。refer page 981

**[DNA催化计算加密 REST 会话中 肽元基Token生成](https://www.zhihu.com/zvideo/1330485210245173248" \t "https://zhuanlan.zhihu.com/p/_blank)**

[](https://www.zhihu.com/zvideo/1330485210245173248" \t "https://zhuanlan.zhihu.com/p/_blank)

[Alkaid 罗瑶光的视频](https://www.zhihu.com/zvideo/1330485210245173248" \t "https://zhuanlan.zhihu.com/p/_blank)

[· 22 播放](https://www.zhihu.com/zvideo/1330485210245173248" \t "https://zhuanlan.zhihu.com/p/_blank)

**[pow（2的60次方，元基长度 ）概率随机DNA加密](https://www.zhihu.com/zvideo/1328926594035953664" \t "https://zhuanlan.zhihu.com/p/_blank)**

[](https://www.zhihu.com/zvideo/1328926594035953664" \t "https://zhuanlan.zhihu.com/p/_blank)

[Alkaid 罗瑶光的视频](https://www.zhihu.com/zvideo/1328926594035953664" \t "https://zhuanlan.zhihu.com/p/_blank)

[· 120 播放](https://www.zhihu.com/zvideo/1328926594035953664" \t "https://zhuanlan.zhihu.com/p/_blank)

**DNA元基隐写术，**

1 DNA元基隐写术的特点是数据隐藏。 作者给大家一个闪光发散点：行为隐写术。refer page 1015

2 设计情报学和安全学，本书不做描述。 refer page

**DNA元基特征，**

1 DNA元基腐蚀特征，可类比生化的DNA基元 如基因。refer page 第7， 8， 9章 元基编码

2 DNA元基物理特征，可类比电工的信号单元 如锁存器等。refer page 第13章 元基数字逻辑

3 DNA元基语义特征，可类比人类的词汇表达，如近义词扩展。refer page 901， 683，

4 DNA元基加密特征，可类比密码学的符号。refer page 901，

**Web登陆token，**

1 Web登陆token 包含物理加密，语义加密，肽展加密，丝化加密，和概率钥匙。refer page 981~结果

2 物理加密 通过物理算法将原文进行元基加密的过程。refer page 981~结果

3 语义加密 通过 语义PDC三元根字典编码 将原文进行元基加密的过程。refer page 981~结果

4 肽展加密 通过 肽展公式进行 元基肽展变换 将原文进行元基加密的过程。refer page 981~结果

5 丝化加密 通过 肽展公式进行 元基丝化变换 将原文进行元基加密的过程。refer page 981~结果

**Session会话加密，**



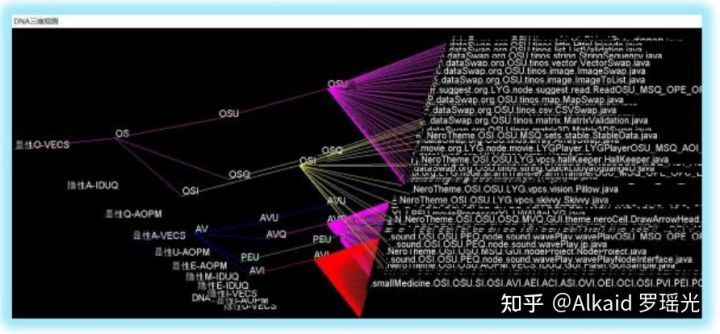
1 Web登陆token 封装的过程 为 Session会话加密.refer page 下册631

2 Session会话加密记录了token加密过程的 概率钥匙记录.refer page 下册631

3 Session会话加密 是一种非对称加密的应用。refer page 下册638

4 概率钥匙的分发方式可以进行 有效的实现 非对称组合加密 应用场景。refer page 下册638

**元基索引**



**[优酷视频](https://link.zhihu.com/?target=https://v.youku.com/v_show/id_XNDkzNTg4OTM4OA==.html" \t "https://zhuanlan.zhihu.com/p/_blank)**

**[养疗经 软件 函数 肽化 染色体索引观测](https://www.zhihu.com/zvideo/1314493149612593152" \t "https://zhuanlan.zhihu.com/p/_blank)**

[](https://www.zhihu.com/zvideo/1314493149612593152" \t "https://zhuanlan.zhihu.com/p/_blank)

[Alkaid 罗瑶光的视频](https://www.zhihu.com/zvideo/1314493149612593152" \t "https://zhuanlan.zhihu.com/p/_blank)

[· 27 播放](https://www.zhihu.com/zvideo/1314493149612593152" \t "https://zhuanlan.zhihu.com/p/_blank)

1 元基索引 最早作者只是 对工程的文件名索引，方便分类.refer page 1015

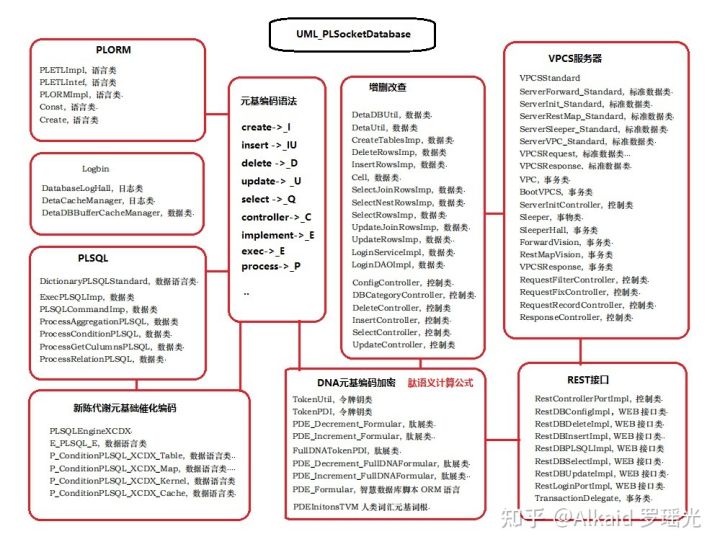
2 元基索引 逐渐进行染色体分类观测，确定函数文件的具体属性。refer page 1015

3 元基索引 开始对函数的具体作用进行特征表达，为了更好的序列化染色体。 refer page 见元基索引花

4 元基索引 用于元基花遗传编码。refer page 下册630

**应用**

数据库加密的文件名 元基索引 新陈代谢. refer page 1018~



**章节的著作权文件列表：**

**第十三章\_DNA\_数术推导与RNA\_X\_THF\_DD元基芯片与肽逻辑**

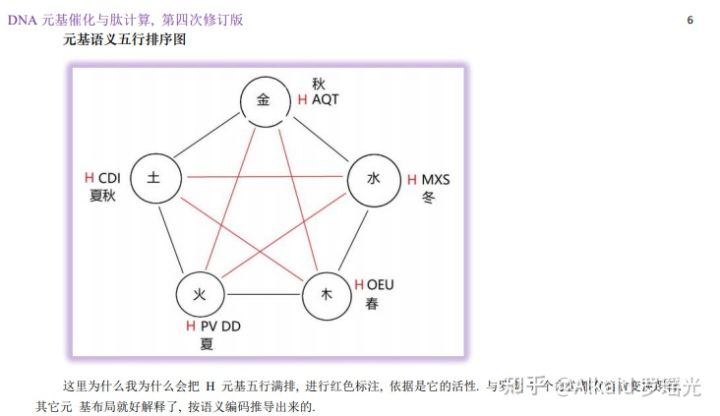
**理解这章节，首先要先进行元基的 O N的数量来进行统计 酸碱腐蚀度比值，进行罗盘归纳元基的活性，和元基的酸碱腐蚀性，主要体现在元基的生化和语义的两种方位排列方式，如下面的罗盘例子展示。作者罗瑶光**

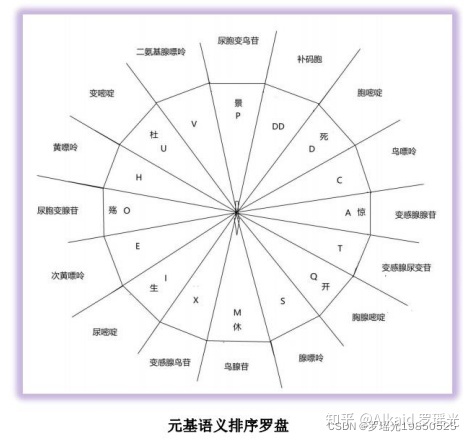


《DNA 元基催化与肽计算 第四修订版V00919》 下册第5页 作者罗瑶光

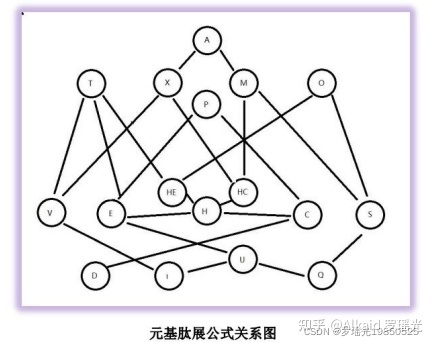
**元基罗盘分类，**

1 DNA元基语义罗盘，refer page 下册6

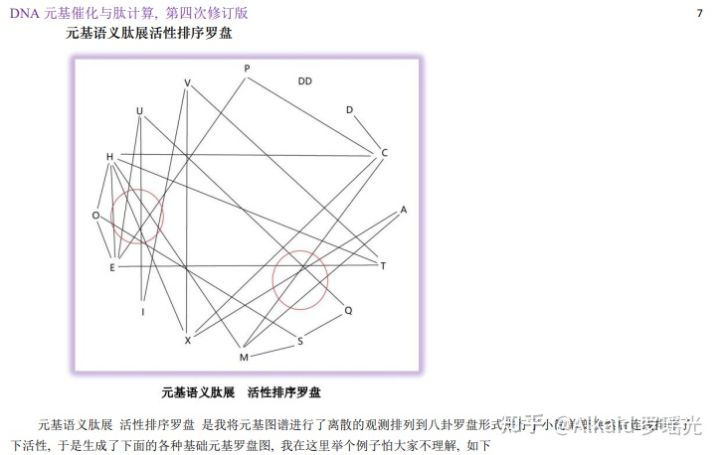




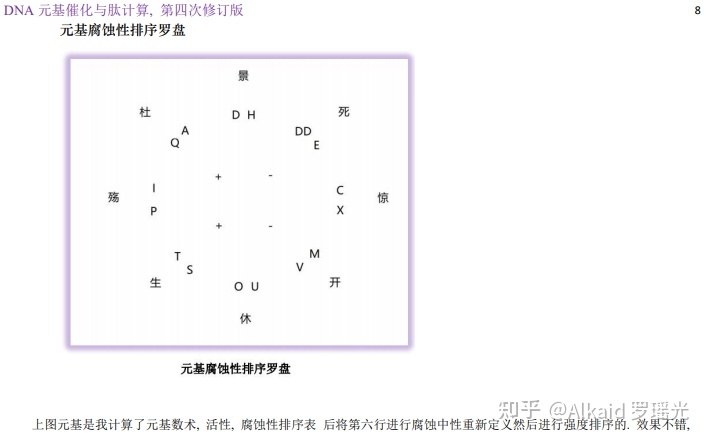
2 DNA元基活性罗盘，refer page 下册6



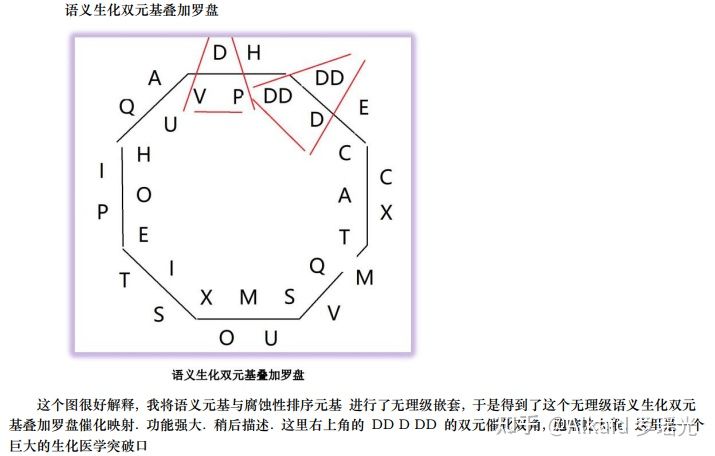
上图进行邻接变换观测如下图



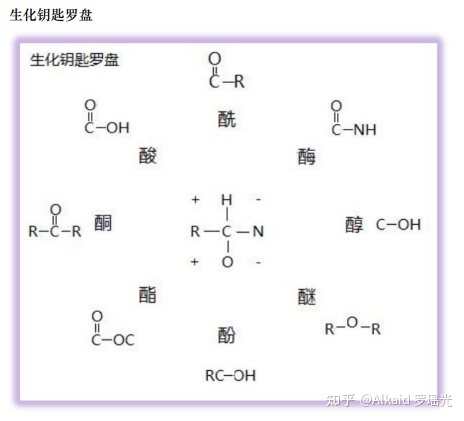
3 DNA元基腐蚀罗盘 refer page 下册7



4 DNA 双元罗盘，refer page 下册8



5 DNA 生化钥匙罗盘，refer page 下册9



6 DNA 语义钥匙罗盘，refer page 下册10



/\* 并入进来。

**肽钥匙，**

1 DNA非卷积视觉的肽钥匙采用化学的 酸酚酮酯 醇酶酰醚 来做钥匙refer page 下册10

2 DNA非卷积视觉的肽钥匙按CNO比例和活性来罗盘归纳refer page 下册9，下册10

3 DNA非卷积视觉的肽钥匙通过罗盘的方位和活性确定其语义属性refer page 下册10

4 DNA非卷积视觉的肽钥匙具备双元 生化语义无理级价值。refer page 下册10



**肽活性表达，略**

1 DNA非卷积视觉的肽元基有化学活性归纳

2 DNA非卷积视觉的肽元基有方位语义归纳

3 DNA非卷积视觉的肽元基有活性归纳

\*/

**元基进制推导，**

1 欧拉计算refer page 下册56

2 商旅分析refer page 下册56

3 十七进制refer page 下册15

4 十六进制refer page 下册16

**十六进制变换方式，**

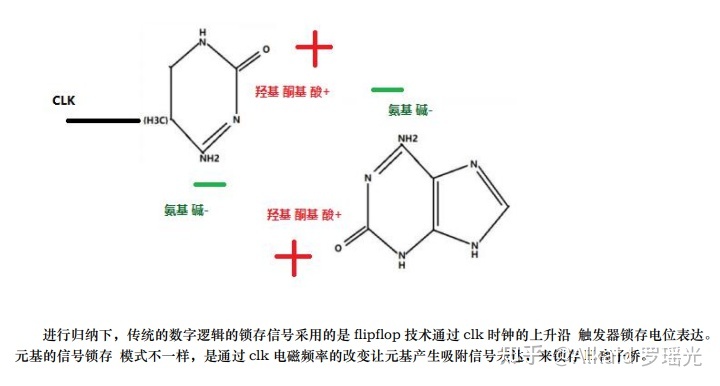
1 十六进制定义 refer page 下册48

**作者因为发现了全嘌呤F于是进行DD 和 HE HC 元基替换观测，发现了DCPE THOS MAXF VIUQ 十六元基欧拉排列，因为首尾是 DQ ，于是定义为人类史第一次定义元基十六进制。以后会不断优化**

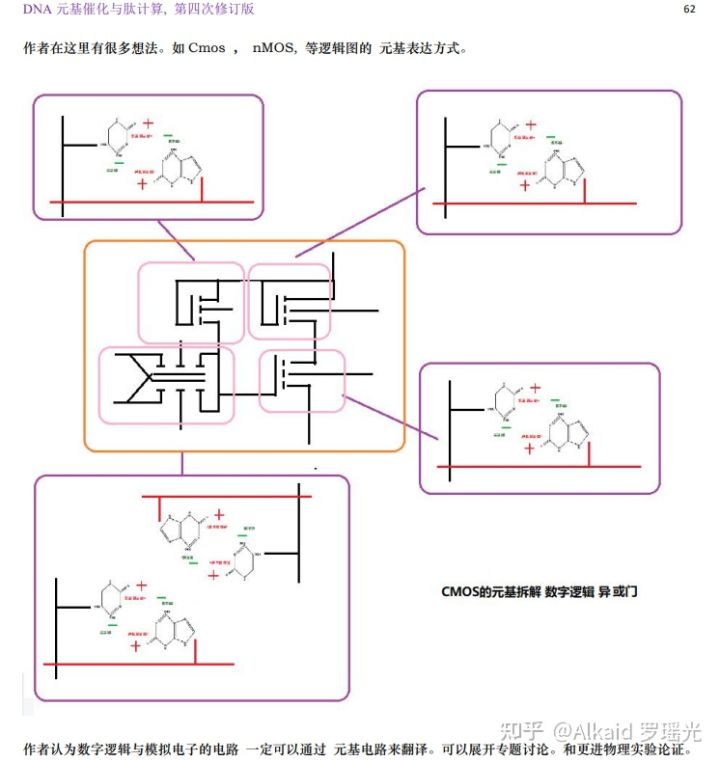
2 十进制互换 refer page 下册11

**元基数字逻辑，**

1 锁存器 refer page 下册60

元基思维发散： 锁存器

2 触发器 refer page 下册61,下册62

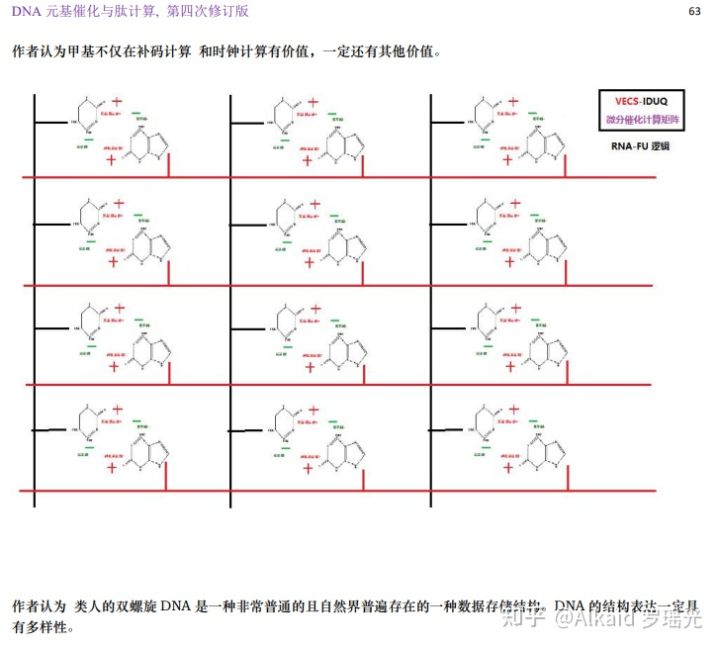
CMOS 元基数字逻辑思维发散

3 寄存器 refer page 下册63

4 锁相环 refer page 下册63

**锁存器与触发器的模拟猜想，**

1 锁相环存储 refer page 下册63

元基阵列存储设计 思维发散

2 锁相环计算 refer page 略

3 锁相环滤波 refer page 下册62

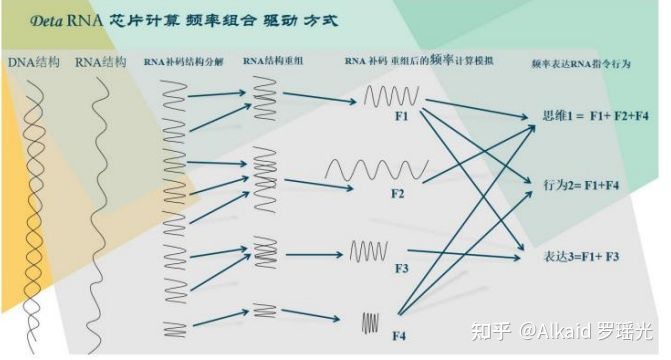
4 计时器 refer page 略

**周期频率语义肽减法公式。**

1 元基频率推导 refer page 下册55

2 元基频率PN极性推导 refer page 下册61

3 元基频率补码减法推导 refer page 下册54



推导过程见书籍《DNA元基催化与肽计算 第四修订版 V00919》

**章节的著作权文件列表：**

**第十四章\_DNA搜索与筛选应用**

这一章节主要涉及 全书的函数进行应用的实践。

**元基的细化模式，**

1 将人类词汇进行语义元基编码。refer page 下册64

2 编码中的元基含有量和元基的搭配位置用于特征标识。refer page 表格中数据元基 在笛卡尔循环搜索时候会自动叠加 略。

3 特征标识主要包含 生化标识和 语义标识。refer page 表格中数据元基 在笛卡尔循环搜索时候会自动叠加 略。

3.1 语义元基定义方式。refer page 下册77

3.2 生化元基定义方式。refer page 下册78

4 特征标识用于搜索和筛选应用。refer page 下册79

**语义的元基表达，**

1 语义的元基表达主要体现在 特征标识的方式。refer page

2 固定的特征标识可以生成元基词汇。refer page 683

3 单个的特征标识可以用于索引分类。refer page 下册79 如筛选分类应用

元基筛选应用实例

4 单位长度的特征标识可以用于索引加密。refer page 下册77 如表格中数据元基描述。

**特征的PCA打分模式，**

1 特征的PCA打分 体现在某元基的 占有概率比重。refer page 下册76

2 特征的PCA打分 体现在某元基团的 占有概率比重。refer page 下册79

3 特征的PCA打分 体现在搜索中权重叠加打分 。举例 张三AOP， 李四POM ,那么 PO 就叠加了，搜索分值权重自动增加。refer page 下册79

**搜索对象的元基索引方式，**

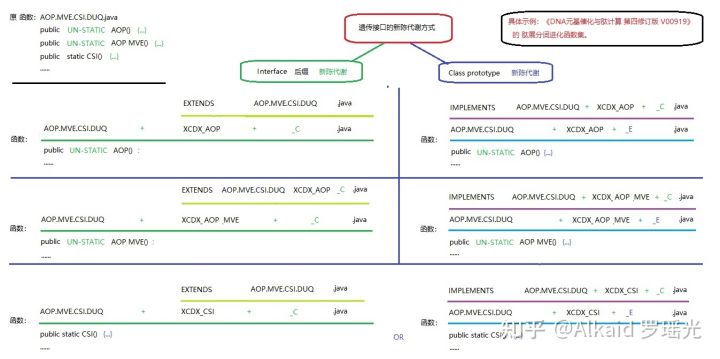
1 搜索对象的元基索引可以通过单个元基染色体分类索引。refer page 下册77

2 搜索对象的元基索引可以通过单个元基词汇 索引。refer page 683

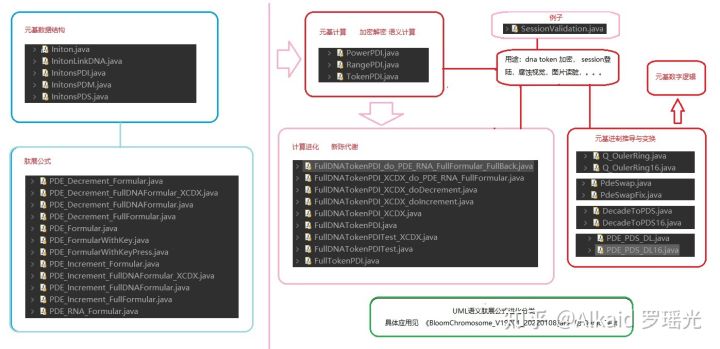
**元基索引染色体分类观测，**

1 元基索引染色体分类观测 体现在函数的功能进行分类。refer page 682，692

2 函数的功能进行元基编码，体现在文件名编码和文件函数名编码。refer page 671，下册147



3 文件名和函数名 元基编码，主要用进行新陈代谢，方便之后的进化计算。refer page 下册149



**应用**

DNN分词词汇花。refer page 下册80~



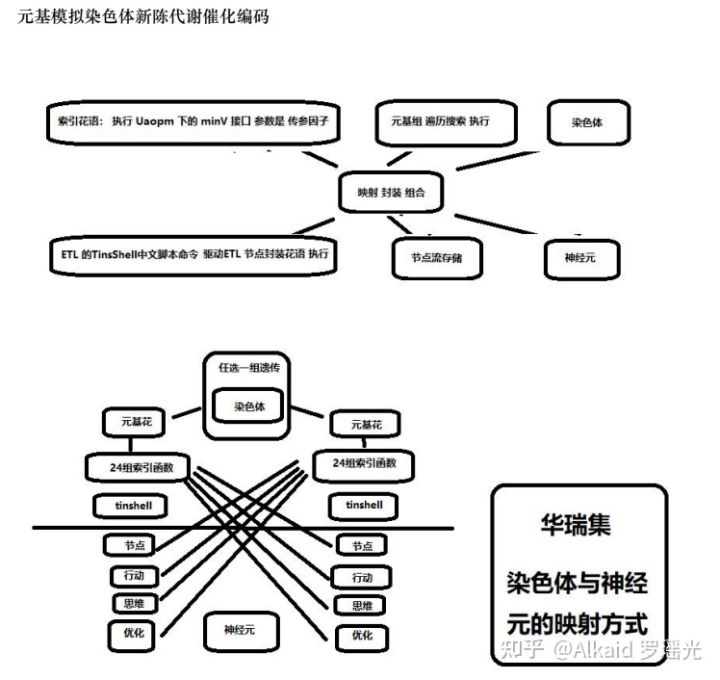
**章节的著作权文件列表：**

第十五章\_元基模拟染色体新陈代谢催化编码

**定义：元基的 新陈代谢一般指 源码工程的函数文件 进行元基编码后的 内容逻辑 接口和类 索引优化方式，主要体现在 文件 的分类，剔除，继承，分配。**

**定义：元基的 二次新陈代谢一般指 源码工程的函数文件名 进行元基编码后的 文件名称 索引优化方式，主要体现在 元基 的分类，剔除，缩进，分配。**

定义者 罗瑶光



**元基造字，**

1 元基造字的编码方式。refer page 672~ ，901~

2 元基造字的编码字典。refer page 901，913

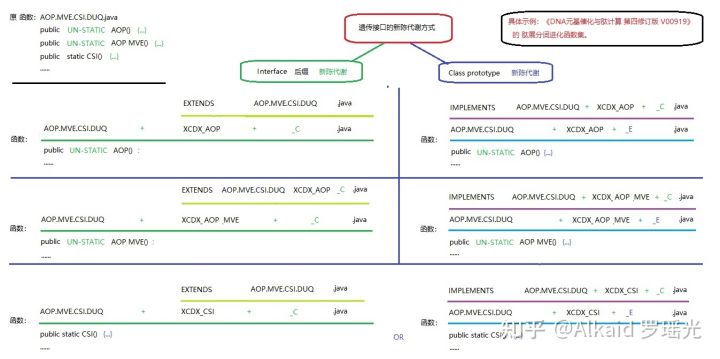
3 元基造字的编码词汇。refer page 语料库方式 914

4 元基造字的字词定义。refer page 下册119（作者的意识而已，没有全民代表性）

**元基进化方式，**

1 元基进化方式 肽展公式新陈代谢。refer page 下册~144~

2 元基进化方式 函数索引二次新陈代谢。refer page 下册149，遗传代谢模式见uml



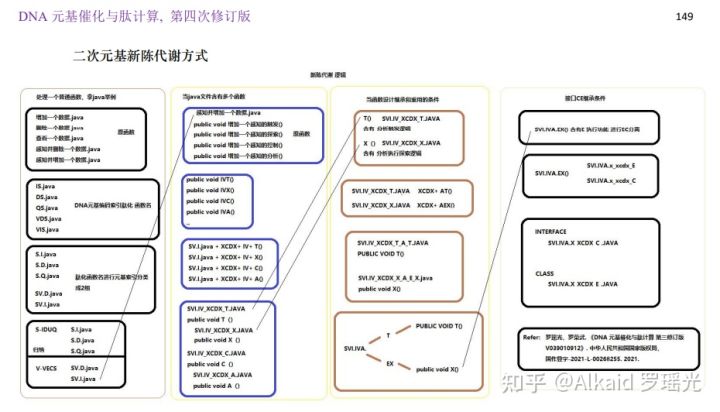
**元基新陈代谢，**

1 文件名新陈。refer page 下册149

2 文件名代谢。refer page 下册149

3 函数名新陈。refer page 略

4 函数名代谢。refer page 略



**元基二次新陈代谢，**

1 文件与函数名的新陈代谢。refer page 下册176~192, 下册214~232, 下册242~274

2 文件内容与函数内容的新陈代谢。refer page 下册172~

3 文件与函数的继承函数新陈代谢。refer page 下册214~274

4 文件与函数的接口函数新陈代谢见CE分离。refer page 下册242,下册248,下册253,下册271

/\*移出本章

refer page 下册172~

**元基花，**

1 元基花染色体模拟。

refer page

2 元基花瓣 映射接口 模拟。refer page

3 元基花萼 接口调用 模拟。refer page

4 元基花蕊 遗传序列 模拟。refer page

**元基枝，**

1 元基枝叶模拟 花蕊集工程文件。refer page

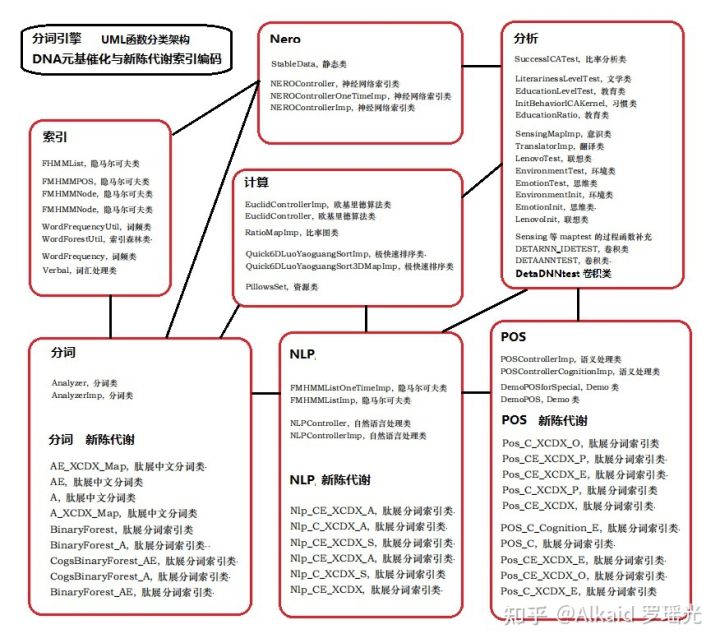
2 元基枝干模拟 养疗经启动文件。refer page

\*/

**元基催化在分词， 排序，图片读脏识别上的应用。**



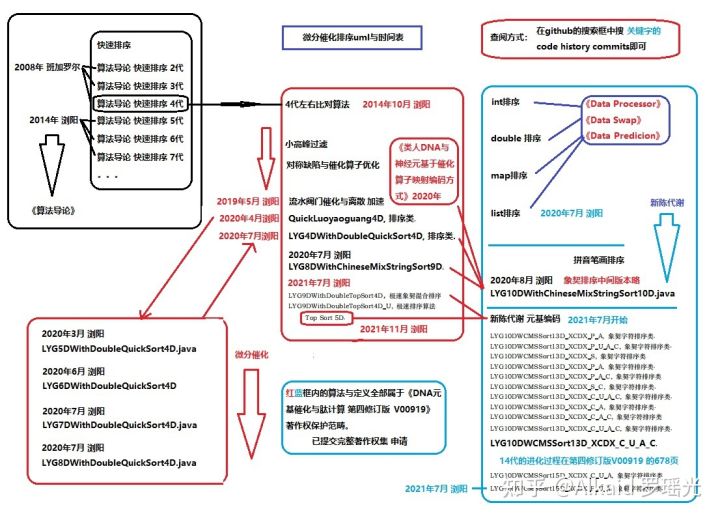
1 肽展催化分词新陈代谢。refer page 下册193~



2 肽展图片读脏新陈代谢。refer page 下册156~



3 肽展象契形排序新陈代谢。refer page 下册172~



**章节的著作权文件列表：**