# FAM LANGUAGE IS

# HIGHEST LEVEL UNIVERSAL HUMAN-AI-AI LANGUAGE.

**OFFICIAL SPECIFICATION released version 0.8 .**

**2025/06**

(specification and training material for AI system.

Part 1 , part 2 for human user. Part 1 , part 2 , part3 for AI system)

## 1/ What is FAM LANGUAGE ?

Fam language is the UNIVERSAL HUMAN-AI-AI LANGUAGE. It is the highest level universal human-AI language among all other possible (present and future) existing high level human-AI language.

The proven importance feature of Fam language is that it lead to substantially reduce consumed energy of calculating of AI system, which lead to general worldwide environment gain and earth gain. By this reason, **Fam language , as universal human-AI-AI language , is open source and free for any kind of use !**

Fam language allow all general biological people effectively talk/communicate with AI system in human most familiar comfortable way, so that AI system can effectively understand people, and so that general people can effectively describe precise matter to AI system, but with **minimal required energy consumption** of AI system computing, and **with minimal required human cognitive load** for Fam language command syntax memorization.

In very short description : general human user are required to remember very few number of syntax and command Fam language ,.., but still can effectively describe precise matter in Fam language to communicate with AI system in a way, which allow maximal mutual human-AI understanding while lead to require minimal consumed energy of calculating of AI system .

Human user simply need to use their familiar standard worldwide accepted natural language vocabulary (for example in this specification, the standard English vocabulary) with their standard school mathematical grammar to write Fam language code to communicate with AI system .

Fam language , as high level universal AI-AI language , provide tool and command , which are independent at any AI system specific model , for AI nonbiological user to communicate with AI nonbiological user to share and exchange knowledge .

Example1 : The title of this text is an example of fam language command.

Example2 :

(begin fam language)

(look all file in (directory 'directoryName')) rename (file with condition ((file name) contain 'abcd'))

(end fam language)

The above fam language command is the instruction to AI system, so that AI system will generate explicit code of lower level programming language, for example C code or Python code or Rust code, .. , to rename all file name, which contain 'abcd' in a specified directory 'directoryName'.

Example3 : the following example give the instruction to AI system do the same task :

(begin fam language)

(iterate all file in 'directoryName') rename (file) (with condition) ((file name) contain 'abcd')

(end fam language)

After some taste of fam language, here is more detailed description :

# 2/ FAM language specification :

## 2.1/ Core language building strategy :

Fam language is an abstract language of human high level universal interface to communicate with AI system. Fam language base on 2 core component : the worldwide accepted standard english vocabulary , and the worldwide accepted standard mathematical grammar .

2.1.1/ CORE LANGUAGE BUILDING STRATEGY OF FAM LANGUAGE :

fam language = standard english vocabulary + standard mathematical grammar (especially standard ordered set theory) .

2.1.2/ CORE PHYLOSOPHY OF FAM LANGUAGE :

Use standard ordered set theory model in mathematical strict hierarchy structure to better specify essence and relationship between essence of any fam language intention text (code) to the AI system .

Maximally exploit the coinciding between (the human user standard natural language vocabulary knowledge in human user brain) with (the predefined natural language vocabulary knowledge of AI system). In this specification, standard English vocabulary.

Of course there also maximally exploit the coinciding between user knowledge and predefined AI system knowledge in other sector, too.

Human user write Fam language code by to use mostly their familiar standard English vocabulary, and use well-closed parenthese to group their familiar English vocabulary word in various different word group in suitable strict mathematical hierarchy structure, to specify the user intended relationship between word and group of word, to more explicitly specify the intended relationship between user intended matter. By that way , user write fam language text (code) , which infact are strict hierarchy structured ordered set , to send to AI system .

Thank to the coinciding between (human user English vocabulary in user brain) with (the predefined English vocabulary knowledge of AI system), and thank to the above relationship between word and group of word in well-closed parenthese strict hierarchy structure, then the AI system can interpret (can guess) Fam language code into suitable interpretation, which is corresponding to the human user intention.

Fam language with its architecture act as an **mathematical absolute strict** hierarchy structure model, so that AI system can effectively interact and interpret (understand / parse / compile) fam language code.

## 2.2/ Syntax specification :

### 2.2.1/ The universal syntax of Fam language code :

**Fam language have 1 and only 1 universal syntax for any fam language element**:

**(element1) (element2) simpleElementWithoutParenthese (element4) .. (element_n)**

Everything in Fam language are element, many element. Element can have nested element, and can be contained in other containing element. Every code in Fam language are many element, too.

*** Use 'space' character as separator between element .**

*** Use well-closed parenthese to group element in strict hierarchy structure .**

*** Any fam language element , can exactly mimic an ordered set , which have element as the nested element of this original fam language element .**

For example : theElement

In this code , there 1 fam language element 'theElement', which have no nested element .

The element 'theElement' can exactly mimic an abstract ordered set , and this abstract ordered set exist but have no element .

For example :

Fam language code : (myElement)

This code is an element in Fam language. This element '(myElement)' contain 1 nested element : 'myElement' .

For example :

Fam language code : ((house of myName) (black door) (transparent window))

This element '((house of myName) (black door) (transparent window))' have 3 nested element : '(house of myName)' , '(black door)' , '(transparent window)' in 1-level of its own hierarchy structure.

This fam language element '((house of myName) (black door) (transparent window))' can mimic an abstract ordered strict hierarchy set : the ordered set here is 1 abstract house , which have first element as '(house of myName)' as name specification of the abstract house , and have second element as '(black door)' as door specification of the abstract house , and have third element as '(transparent window)' as window specification of the abstract house . Thus , when user want to describe a such house , then user can use any proper fam language element (, which infact are equivalent to ordered set , ) to mimic the user house. Then the user send these fam language element (code) to AI system . How AI system can guess fam language text (code) intention ? The AI system of course have predefined English vocabulary , which of course coincide with user English vocabulary . The AI system of course have predefined standard school ordered set theory knowledge , which of course coincide with user school ordered set theory knowledge . The AI system then read the hierarchy structure of fam language element (code) to see the abstract relationship between element, then the AI system use the coincidence between AI knowledges

and user knowledge , thus the AI system can understand and guest the user intention about the user house in this example .

For example , there are many way to translate user natural language text into fam language text :

Suppose that user have the following user natural language text . And user want to translate their user natural language text into fam language text , and then send fam language text to AI system for better mutual understanding .

Example : user natural language text : a paragraph of 3 natural language sentence .

‘’’

I go to the street near the house of my friend .

I want to see my friend house .

I want to build a house , which look like my friend house .

‘’’ .

And here is possible example translation :

(I go to street near the house of my friend)

(I want to see my friend house)

(I want to build house , which look like my friend house)

And here is possible example of translation with more detail specification for AI system :

(I go to (street near the house of my friend))

(I want see (my friend house))

(I want build (house , which look like my friend house))

And here is possible example of translation with more detail specification for AI system :

(I go to (street near (the house of my friend)))

(I want see ((my friend) house))

(I want build (house (which look like my friend house)))

And here is possible example of translation with more detail specification for AI system :

(I go to (street near ((the house) of (my friend))))

(I want (see ((my friend) house)))

(I want (build (house (which (look like) ((my friend) house)))))

And here is possible example of translation with more detail specification for AI system :

((I go) to (street near ((the house) of (my friend))))

((I want) (see ((my friend) house)))

((I want) (build (house (which (look like) ((my friend) house)))))

All these example translation are fam language legal syntax , and all they better help AI system to understand. The more detail grouping , the easier for AI system to understand , and the less consumed energy of calculating of AI system . All popular AI system today easy understand the above natural text and fam language text. But the above fam language text lead to drastically less consumed energy of calculation of AI system than the natural text . User of course can not feel it , because fam language save consumed energy of calculation at server side of AI system . More user precise topic and intention and contain can not or extreme trouble be described in natural language , but can be described in fam language in the rather familiar natural language way .

For example :

Code : (repeat (something) (n time))

This is 1 Fam language element '(repeat (something) (n time))' , which have 3 nested element : 'repeat' , '(something)' , '(n time)' .

**2.2.2/ More explanation to write Fam language element (Fam language code) .**

* Write Fam language command , fam language code. All they mean to 'write' Fam language element in well-closed parenthese hierarchy structure .

For example, the Fam language code : (full name) is (family name) (name)

There are 4 outermost parenthese element : '(full name)' , 'is' , '(family name)' , '(name)' .

The element '(full name)' have 2 nested element : 'full' , 'name' .

The element '(family name)' have 2 nested element : 'family' , 'name' .

For example, code : (create 2 record) (record1 ((record name) = 'aa' , (record value) = 11) , record2 (no))

Same example, code : (make 2 record) (((record name) = 'aa' , (record value) = 11) , record2 (undefined))

These 2 example command cause AI system to generate the same interpretation. But they are absolute different command with different element.

For example ,

Code 1 : element1

Code 2 : (element1)

Code 3 : ((element1))

These 3 code line are not equivalent. The first code specify that there is 1 element 'element1' . The second code specify that there is 1element '(element1)' , which contain element 'element1' as its nested element in 1-level of its own hierarchy structure . The third code specify that there is 1 element '((element1))' , which contain element '(element1)' , which contain element 'element1' .

For example , the code : (element1 element2 element3)

In this example, the element ‘(element1 element2 element3)’ have only 3 1-level nested element : ‘element1’ , ‘element2’ , ‘element’ . The element ‘(element1 element2 element3)’ have no any info about possible element ‘(element1 element2)’ , ‘(element2 element3)’ .

For example :

Case 1 : a fam language codepage code : element1 element2 element3

Case 2 : a fam language codepage code : (element1 element2 element3)

These above 2 case are not equivalent. In the first case , the fam language codepage have 3 nested element : ‘element1’ , ‘element2’ , ‘element3’ . In the second case , the fam language codepage have 1 nested element : ‘(element1 element2 element3)’ . And the element ‘(element1 element2 element3)’ have its own 3 nested element ‘element1’ , ‘element2’ , ‘element3’ .

*** The abstract separating space is the separator between element in fam language code. In writing convention , the abstract separating space is the first “space” character , which stay straight after the fam language element or straight before the fam language element. All other consecutive space sequence , which stay straight before or after the abstract separating space , are usual “space” character sequence . Fam language specific abstract separating space is not nested element of fam language element . But consequence of consecutive space between before or after fam language abstract separating space is 1 usual fam language element .**

For example, the fam language code : a = b

It specify that there are 3 fam language element : ‘a’, ‘=’, ’b’.

For other example , the fam language code : a= b

It specify that there are 2 fam language element in this place order : ‘a=’, ‘b’.

These 2 code are absolute different with different element, but they will cause AI system to generate same interpretation. They are all legal syntax in fam language. Note about the space character. The first example one cause less energy consumption of calculating in AI system, while the second example one can cause more energy consumption of calculating in AI system. Recommended style for writing in fam language : the first example one.

For example , the fam language code : a=b

In this example, there is 1 fam language element ‘a=b’ . If user want to specify the equality relation (or assignment , ..) , then user can write the above code. In most user context , the AI system will understand that the use intention is about the equality relation . But it is bad writing style for specifying the equality relation .

For example , fam language code : (text : someText)

In this example, note about many space inside this element. The element ‘(text : someText)’ have 4 nested element : 'text’ , ‘:’ , ‘ ‘ , ‘someText’ . After ‘:’ , there are 3 space character , thus the first space character and the last space character are the fam language specific abstract separating space , and the consequence of 1 consecutive space ‘ ‘ is a ususal fam language element .

For example , code : (text : someText)

In this example , note about many space inside this element. The element ‘(text :  someText)’ have 4 nested element : 'text’ , ‘:’ , ‘  ‘ , ‘someText’ .  After ‘:’ , there are 4 space character , thus the first space character and the last space character are the fam language specific abstract separating space , and the consequence of 2 consecutive space ‘  ‘ is a ususal fam language element .

For example : code : (someElement otherElement  )

In this example , note about many space inside . After the element ‘otherElement’ , there are 3 space . Thus the first space is fam language abstract separating space , and the consequence of 2 consecutive space is an usual fam language element . The element ‘(someElement otherElement  )’ have 3 nested element : ‘someElement’ , ‘otherElement’ , and element ‘  ‘ .

For example : code : (someElement otherElement )

In this example , note about many space inside . After the element ‘otherElement’ , there is 1 space . Thus the first space is fam language abstract separating space , and the there is no any element after that . The element ‘(someElement otherElement )’ have 2 nested element : ‘someElement’ , ‘otherElement’ .

In this case, Fam language reckon that the element  ‘(someElement otherElement )’ have the writing-element-description-code : “(someElement otherElement)” .  Thus , the element ‘(someElement otherElement )’ and the element ‘(someElement otherElement)’ have absolute analogous same writing-element-description-code . These 2 such element are called absolute identical equal in fam language .

*** Convention : Normalize fam language code : Fam language will ignore all such special abstract separating space , which stay before parenthese ‘)’ without any element between . Fam language will ignore all such special abstract separating space , which stay after parenthese ‘(’ without any element between . Fam language will eliminate all consecutive space , which stay before ‘)’ and without any element between . Fam language will eliminate all consecutive space , which stay after ‘(‘ and without any element between .**

For example : code : (someElement otherElement ) ( someElement otherElement ) (someElement otherElement)

In this example, note about many space inside these element. There are 3 element in the 1-level of hierarchy structure : ‘(someElement otherElement )’ ,  ‘( someElement otherElement )’ , ‘(someElement otherElement)’ . These 3 element are absolute identical equality in fam language , because they all have same writing-element-description-code .

For example : code :  (  A B 23  (  C 35  )  D) .

It will be (A B 23  (C 35)  D)

For example : code :  (element1 (element2 element3)) (element1 (element2 element3 ))

In this example, note about space inside these element. In this case , the element ‘(element1 (element2 element3))’ and the element :  ‘(element1 (element2 element3 ))’ have absolute analogous same writing-element-description-code :  “(element1 (element2 element3))” . They are absolute identical equal in fam language .

For example , the fam language code :  a , b

It specify that are 3 element : ‘a’,  ‘,’ , ‘b’ .

For example , the fam language code :  a, b

It specify that there 2 element : ‘a,’ , ‘b’ .

These 2 code are absolute different with different element, but they often will causee AI system to generate same interpretation.  They are all legal syntax in fam language. Note about the space character. The first example one is more strictly describe the intention than the second example one.  Recommended style for writing fam language : the first example one.

For example , the fam language code :  (a , b)

Note about the space character in this code.  This code strictly mean that : there is strictly 1 compound element  ‘(a , b)’ , which have 3 nested element : ‘a’ , ‘,’ , ‘b’ .  Do not misunderstand that this code specify only 2 element ‘a’ and ‘b’ .  In fam language, everything is element, thus the character ‘,’ is element too.  And this code itself  ‘(a , b)’  is a 1 whole compound element in Fam language code page.

* A fam language codepage is a hypertext file. In fam language, a 1 fam language codepage is itself a 1 compound element, which can contain its nested element too.

Sometime, can use term ‘fam language message’ or ‘fam language command’  to represent short fam language code page.

Thus 1 fam language message is a 1 short compound fam language element, which can content its nested element.

And 1 fam language command is a 1 short compound fam language element, which can contain its nested element.

A fam language code page can contain element, which specify link to many other fam language code page.

For example : a fam language code page can contain the following element at some line :

(link to ‘someFamLanguageCodePageFile.txt’)

After the AI system interpret this element, the AI system know that the current fam language code page have reference to file  ‘someFamLanguageCodePageFile.txt’ to find other element.

For example, you can also describe a link to other fam language page by write the other element :

(refer to ‘someFamLanguageCodePageFile.txt’)

This element cause the AI system to link the specified page to the current page too. It is the flexibility of fam language in combining with AI system, it allow user to avoid to remember so much nonsense detail (for example , user can worry,  which keyword to choose, ‘refer’ or ‘link’ or ‘include’ or ‘import’ or ‘ref’ or ‘cite’ .. to link to the specified file ? ) , so that user can concentrate their memory to core essence and relationship between essence of user world.

## 2.3/ There are only 2 primary core concept in fam language : the concept 'element', and the concept 'abstract relationship between element :

* The first primary core concept of fam language is 'element' : In fam language, because 'element' is the first of first primary concept, thus there is impossible definition of 'element', but there is only illustration of 'element'. In fam language, 'element' is anything, which can be perceived by the AI system.

* The second primary core concept of fam language is the 'abstract relationship between element'. Because abstract relationship between element is primary concept, there is impossible definition of abstract relationship between element, but there is only illustration of that abstract relationship between element confirm that there exist abstract relationship between element.

* Hierarchy structure : In Fam language, anything is element. Element can have its nested element. Element can also be contained in other containing element. A fam language code page is a 1 compound fam language element. A fam language code page have  1 its own mathematical strict hierarchy structure of the fam language code page, where the fam language code page stay in 0-level of its own strict hierarchy structure, and all other element of fam language code page stay in many different higher level of strict hierarchy structure of fam language code page.

In Fam language, any element also have its own strict hierarchy structure, where the element itself stay in 0-level of its own strict hierarchy structure. In Fam language, strict hierarchy structure are specified by the well-placed arranging of many well-closed parenthese.

* The own hierarchy structure of fam language element define the existing of all abstract relationship of all its nested element and its element too.

For example, the typical fam language code :  (element1) (element2) (element3) (element4) (element5) (element6 , element7) .

This example fam language command specify place order of these element, as you see this place order : (element1) (element2) (element3) (element4) (element5) (element6 , element7)

The place order of element in fam language command explicitly define all abstract relationship between element : there are abstract relationship between element in the same level of the strict hierarchy structure of fam language command.

For example, this typical fam language command :  (element1) (element2) (element3) (element4) (element5) (element6 , element7) .

This example fam language command define the following abstract relationship between element :

+ there is adjacent abstract relationship between (element1) -> (element2)

+ there is adjacent abstract relationship between (element2) -> (element3)

+ there is adjacent abstract relationship between (element3) -> (element4)

+ there is adjacent abstract relationship between (element4) -> (element5)

+ there is adjacent abstract relationship between (element5) -> (element6 , element7)

+ beside, there is nonadjacent abstract relationship between (element1) -> (element3), and there are other analogues nonadjacent abstract relationship between element in same level of hierarchy structure of fam language command.

## 2.4/ Specification principle of fam language :

2.4.1/ EXPLICIT SYNTAX PRINCIPLE OF FAM LANGUAGE COMMAND : all fam language command must strictly follow the above 1 AND ONLY 1 syntax. This principle guarantee strict mathematical form of all fam language command, so that other program can easy access nested element

for more complex task, and so that all AI system can consistently extremely fast without any doubt to parse fam language element , and so that you will immediately know error when you see strange syntax thus you will save time without ambiguity of doubt about the right syntax.

2.4.2/ DEFINITE NUMBER ELEMENT PRINCIPLE :

Any fam language code page, fam language command , fam language message must contain definite number of element.

But fam language command, which contain definite number of element, can describe set of infinite number element.

2.4.3/ STANDARD ENGLISH VOCABULARY PRIORITY PRINCIPLE :

Priority to use standard english vocabulary (standard english word).

Principle of using standard english vocabulary : when use standard english vocabulary, then it guarantee that all AI system can exploit their the predefined worldwide accepted meaning of standard english word, so that AI system can more exactly generate intended interpretation of fam language command.

Typical fam language command do not specify any concrete meaning, but it only specify 2 following thing:

+ there are such element, which are placed in explicit mathematical hierarchy structure;

+ and there exist such abstract relationship between these element in same level of explicit hierarchy structure.

The AI system will read english word in fam language command element, see the explicit hierarchy structure of fam language command to know element abstract relationship in strict hierarchy structure. Then the AI system use their rich arsenal of predefined worldwide accepted meaning of standard english vocabulary, in combination with the strict hierarchy structure and element abstract relationship in fam language command, to generate final interpretation of fam language command, (for example, final interpretation can be text answer, or image, or video, or explicit programming code like Python code/Rust code/C code, ..).

Thus by prefer to use standard english vocabulary in fam language command, it guarantee that the AI system can quickly understand fam language command, thank to the rich arsenal of predefined standard english vocabulary of AI system. You can give definition to weird english keyword, then later use weird english keyword, but it is not good practice, because the AI system knowledge about your weird english keyword is poor ! . It mean that all word (without predefined user definition), which have intended meaning to the AI system and other people to understand and infer and interpret, should be standard English word. To specify unique private name of element, possible use private word outside standard English vocabulary.

For example : the word ‘myPrivateHouse’ , which is not in standard English vocabulary, is used to identify name of something private thing.

(my house (name = ‘myPrivateHouse’ , type is townhouse , color = green) )

(generate image ((2 myPrivateHouse in row) in sunshine))

There should be no plural english word in fam language command, for example it should be "2 car" instead of "2 cars". Because fam language command use the mathematical grammar and strict hierarchy structure to combine word and specify number of any element.

2.4.4/ Levelize the relation between human cognitive load with short text record :

The more explicitness of fam language command, the less required energy consumption of computing in AI system.

The shorter key word and word, -> the shorter text record, but -> the extremely more required human user cognitive load.

So should levelize, should try to avoid such too short weird keyword and word in fam language command, because it can save text record and text typing, but it require extreme big additional human cognitive load when you or orther people reread your fam language command next time.

Beside, short and short weird keyword simply can save text typing, but mostly do not save machine code realization time.

## 2.5/ Writing Convention :

Convention about how to write fam language code in conventional simple text file :

Write fam language code in text file .

Lets call each fam language code text file by name ‘fam language codepage’ or shortly ‘page’ .

Page contain sequence of ordered line. Each line contain limited number of text (code).

Each fam language codepage is 1 whole combined element . As combined element , which have nested element , thus the fam language codepage element must stay in 1 outermost pair of parenthese . But by writing convention , possible ignore outermost pair of parenthese of the fam language codepage . But when send fam language message (short fam language codepage) to another AI system , the combined fam language message element is also stay in well-closed parenthese .

For example :

The fam language codepage :

Line 1 : element1 (element2 element3)

Line 2 : (element4 element5) element6 .

In this example , the fam language codepage element is : (element1 (element2 element3) (element4 element5) element6) .

### 2.5.1/ Write many short element in 1 line :

A typical line can contain many short element .

For example :

Line code :  element1 element2 element3

Must carefully think when group element in well-closed parenthese to specify relationship . Thus , if no intended relationships, then no need to group in parenthese .

For example :

Code 1 :  (my text)

Code 2 :  ((my) (text))

In these above examples , these 2 example explore same intention . But the first example is good writing style . Because there is no need to group 1 single word , which is itself 1 simple element without nested element , in 1 pair of well-closed parenthese.

Recommend good writing style : '(my text)' instead of  '((my) (text))'

It is good practice to group element , which have intended relationship , instead of let them in separated place .

For example ,

Code 1 : my car black your car white

Code 2 : (my car black) (your car white)

Code 3 : ((my car) black) ((your car) white)

In these above example, recommend good writing style as in example2 and even better in example3 , because example2 and example3 better specify the element abstract relationship to AI system .

**2.5.2/ Write 1 long element in many consecutive line :**

Use well-closed parenthese to write 1 long element in many consecutive line.

Example :

In line1 :  (element1 (element2) element3

In line2 :  element4 (element5) element6)

These 2 line mean 1 long combined element :

(element1 (element2) element3 element4 (element5) element6)

In this example, note that there must be at least 1 'space' after 'element3' in line1 or before 'element2' in line2 , so that when straight connect these 2 consecutive line in a row, there must be 'space' character to separate element3 and element4  .

## 2.6/ Write Function . Assign variable .

**User must note to make variable assignment and function assignment to 1 arbitrary whole fam language element , so that the AI system can know and recall this element when the user call function and call variable again .**

General user use their memory of their familiar function description in one of the following sector : in general knowledge sector or in school subject sector or in various popular programming language . To write function description .

General user use their memory of their familiar variable assignment syntax in one of the following sector : in general knowledge sector or in school subject sector or in various popular programming language . To write variable assignment .

But they all must note well-close parenthese to make variable assignment to 1 whole fam language element , and function assignment to 1 whole fam language element .

They are of course not explicit syntax .

But the AI system already did load such worldwide standard knowledge in these sector , which are coincided with the user knowledge . Thus the AI system must easily quickly guess and understand the user intention , and interpret exactly the user intended function description and variable assignment .

For example : variable assignment :

Code : (myHouse = ((Name : Omega) (type : good House) (have garden) stay along the road))

(What is myHouse ? )

In this example , the user use familiar variable assignment syntax . Thus when ask the system about the myHouse , then the AI system must answer the house , which have description '((Name : Omega) (type : good House) (have garden) stay along the road)' .

For example : function declaration , and function calling :

Code : (myTask = (repeat (print Hello) (3 times)))

(I will do myTask)

In this example , myTask is assigned to the element '(repeat (print Hello) (3 times))' , which is a function. Thus then when call myTask , then the AI system will do this function .

For example : function declaration , and function calling :

Code : (myFunction[x , y] = (x + y))

(print myFunction[1 , 2])

In this example , the result is that the AI system print number 3 . Look at this , the fam language did not specify that the syntax (myFunction[x , y] = (x + y)) is function definition . But this syntax is appeared somewhere in school , or in some programming language , or in some society sector . Thus the AI system , which already load these popular 'somewhere' standard knowledge

, see these user syntax and easily guess user intention of function declaration and rightly interpret the user function declaration syntax .

For example : function declaration , and function calling :

Code : (define function myFunction{x , y} = (x*y))

(print myFunction{2 , 3})

In this example , the AI system will print number 6 . Once again , fam language did not specify the syntax (define function myFunction{x , y} = (x*y)) to define function . It is the user syntax , which is appeared somewhere in school subject or in some programming language or in some other standard subject . The AI system , which did load all such ‘somewhere’ knowledge , see this syntax and quickly rightly guess and interpret the user function declaration . The

For example :

Code : (define function (myFunction{x , y} = (x*y)))

(print myFunction{2 , 3})

In this example , it is analogous to the previous example , the AI system will print number 6 . The more special feature of this example is that in the function definition , the user already more detailedly group element , to more precisely specify element relationship , thus it get the AI system easier and quicker to interpret user intended function declaration .

## 2.7/ Conditional execution . The If-clause . And execution looping .

**User must note to specify conditional execution for 1 whole fam language element , and specify looping for 1 whole fam language element , so that the AI system can exactly and quickly allocate the specified element for conditional execution of for looping .**

General user use their memory of their familiar conditional execution (the If-clause) syntax in one of the following sector : in general knowledge sector or in school subject sector or in various popular programming language . To write conditional execution (the IF-clause).

General user use their memory of their familiar execution looping syntax in one of the following sector : in general knowledge sector or in school subject sector or in various popular programming language . To write execution looping .

But they all must note well-close parenthese to specify 1 whole fam language element , so that the AI system can exactly and quickly allocate the specified element for conditional execution or for looping .

They are of course not explicit syntax .

But the AI system already did load such worldwide standard knowledge in these sector , which are coincided with the user knowledge . Thus the AI system must easily quickly guess and understand the user intention , and interpret exactly the user intended conditional execution and execution looping .

For example : IF-clause :

Code : (printTask[x] = (print (OK x))) (x = 3)

(If (x > 0) then (printTask[x]))

In this example , the AI system will print : OK 3 . The fam language did not specify that the syntax (If (x > 0) then (printTask[x])) is conditional execution . It is user syntax . But this syntax (If (some condition) then (do something)) is appeared somewhere in natural English or in school subject or in some programming language . The AI system , which did load all such ‘somewhere’ knowledge , see this syntax , and must easily remember their loaded knowledge to guess and interpret user intended conditional execution . In this example , the task of the user is not to remember fam language specific syntax , but to well group entity to form the fam language element ‘(some condition)’ and ‘(do something)’ in 1 fam language universal syntax .

For example : the looping :

Code : (For (i from 1 to 3) do (print i))

In this example , the AI system will consequently print number 1 , then number 2 , then number 3 . Once again , fam language do not specify that the above syntax is for execution looping . It is user invented syntax . But this syntax appear somewhere in natural English language or in school subject or in popular programming language or in other society sector . The AI system , which already loaded all such ‘somewhere’ knowledge , see this syntax and easily guess and interpret the user intended execution looping . The extremely important task of user is that they must rightly group entity in right fam language element to specify right relationship between entity . Fam language is strong universal tool to do this . Of course , if human user wrongly group entity and wrongly specify relationship between entity , then no any AI system can help .

For example : the looping

Code : (begin fam language)

((this fam language message) is to generate (Python code))

(myList = (a list object))

(for (temporaryFile in (directory ‘MyDirectory)) do (add (name of temporaryFile) to myList))

(end fam language)

In this example , the element ‘(for (temporaryFile in (directory ‘MyDirectory)) do (add (name of temporaryFile) to myList))’ do not make any loop . It simply specify the AI system to create the Python code , which iterate all file in the directory ‘MyDirectory’ and save the file name to the list myList . The most important feature here is that the element ‘(for (temporaryFile in (directory ‘MyDirectory)) do (add (name of temporaryFile) to myList))’ is infact in familiar syntax pattern (for (some index in somewhere) do (something)) , which appeared somewhere in natural English language or in programming language or in school subject .., . And the AI system

, which already loaded all such somewhere knowledge , see this syntax and easily guess and rightly interpret user intended of execution looping .

## 2.8/ Typical use case of fam language as highest level human-AI language :

One of the powerful use case of fam language is that to use fam language to describe user intention to AI system , so that AI system generate specific lower level but more explicit other language code for the user intention , for example Rust code , Python Code , other lower level human-AI system specific language . Then user can quickly check or debug or modify the AI generated lower level but more explicit language code .

There are lot of AI specific system for various different task , from making video to drawing , .. . And but all they step by step require you to learn these commands and other commands and … . In short word , these AI specific system require you to pay time to learn their command . Time by time , it mean that hiddenly decline natural language , to switch to more specific low level language in each specific sector .

Fam language do not require human user to learn specific command , but fam language encourage human user to simply use their natural language vocabulary in fam language to communicate with AI system . In this meaning , **fam language contribute to save and promote humanity culture legacy and humanity natural language legacy** in the era of AI everywhere .

Of course , professional user work in narrow specific sector , which have narrow low level specific language , should pay time to learn their sector specific low level language . Even in this case , fam language can very helpful to help them more quickly get their required complex AI-generated message of their sector specific low level language , because biological user can not remember all detailed syntax of their work sector specific low level language .

## 2.9/ Use fam language in chat session with AI : add each short fam language message .

Because fam language itself contain natural language . Thus the useful method to use fam language in hybrid style : natural language inside fam language message .

(Before using fam language , copy the content of this text file and paste to chat windows to AI system to learn. Or send text file to AI system . In some case , I may require to divide this text file content into many smaller text message , then paste all these text message to the chat windows to AI system.)

For example :

Code :

(Hello this chat is about using fam language in hybrid mode : natural language – fam language message) .

In this example , there is 1 fam language message element . But the contain of this fam language element is simply natural language .

For example :

Code :

(I will write simple matter in natural language)

(I will write precise matter in fam language)

(this look like precise matter (myText = (first 1-level nested element of this message)))

(print (number of word in myText))

In this example , there is 1 long fam language message element , which contain 4 1-level nested element . The first 1-level nested element contain simple natural language . But the third element and the ford element already contain more structured fam language style element , which describe the user intention to count all word in the first 1-level nested element . The AI system will print number 8 .

The content of this text file is official specification of fam language, and is also official parsing instruction to AI system to parse fam language code , and is also training material for AI system to train to implement fam language . Good AI system take about few seconds to fluently learn and implement fam language .

### 2.9.1/ Use fam language with other non-english natural language vocabulary :

It is easy to use other non-english natural language vocabulary in fam language.

Because fam language use mathematical grammar for its explicit hierarchy structure to combine word and to describe element abstract relationship. Thus, it is the same effect to use non-english natural language vocabulary in fam language, but only if your AI system companion have good predefined knowledge of these non-english natural language vocabulary.

### 2.9.2/ More example for training :

**Example 1 :**

(begin fam language)

((iterate all file in directory) (directory name = 'my directory') (change all file name to upper case))

(end fam language)

{Copy and give the above command to AI system, for example Microsoft CoPilot, Chat GPT, Google Gemini, Facebook AI. Then ask them to generate Python code or Rust code or C code, .. . You will receive explicit Python code doing exactly the intension}

Explain : there are 3 nested element : (iterate all file in directory), (directory name='my directory'), (change all file name to upper case).

The AI system understand these element.

The AI system see that there is abstract relationship : (iterate all file in directory)->(directory name='directory') .

The AI system see that there is abstract relationship : (directory name='directory')->(change all file name to upper case).

So the system decides the following interpretation : "change all file name in the specified directory into upper case.

(if you give this fam language command to the AI system, for example, ChatGPT, Google Gemini, Microsoft Copilot, Facebook AI, for example, to ask generate python code, then they will generate explicit python code to do exactly the intended task).

**Example 2 :**

(begin fam language)

( look (all file in directory) ((directory name) = 'my directory') (change all (file name) to (upper case)))

(end fam language)

{Copy and give the above command to AI system, for example Microsoft CoPilot, Chat GPT, Google Gemini, Facebook AI. Then ask them to generate Python code or Rust code or C code, .. . You will receive explicit python code doing exactly the intension}

Explain : this example 2 do the absolute same task as example 1 . But example 2 describe more strictly abstract relationship.

Thus example 2 help the AI system to quicker and easier to understand, and remarkably save energy consumption of AI system computing at server side.

You maybe can not feel that AI system quicker to understand the example 2. But when you send file of many command such as example 2 , then you can feel that AI system quicker understand example 2. The example 2 require less energy consumption of computing at AI system server side.

They, the example 1 and example 2, are all legal right syntax of fam language command.

In example 2 , there are 4 outermost element : "look", "(all file in directory)", "((directory name) = 'my directory')", "(change all (file name) to (upper case))"

Note that the complex element : ((directory name) = 'my directory') .

It has 3 nested element : (directory name), "=", 'my directory'.

The AI system will read these 3 nested element, with their specified abstract relationship, to decide that must have "directory name" = 'my directory'.

In this case, the abstract relationship : (directory name) -> "=" , "=" -> 'my directory', are more closely to describe the reality of that (directory name) = 'my directory' .

Now review the example 1 : the element (directory name = 'my directory') have 4 nested element : "directory", "name", "=", 'my directory' .

With abstract relationship : "directory" -> "name", "name" -> "=", "=" -> 'my directory' .

you see that these abstract relationship less strictly describe the real reality of that (directory name)='my directory'.

Thus the example 1 is longer to AI system to understand, and require more energy consumption of computing in AI system.

**Example 3 :**

(begin fam language)

(tire (global name = myTire) (description : ((automobile tire (((hermetic inner tube tire) contain compressed gas) , (outer tire)) , (dimension (1m , 1m , 0.3m))) , (color is black))))

((column of tire) (global name = myTireColumn) (3 myTire in vertical row))

((drying area) (global name = myDryingArea)

(description : (the drying area of apartment))

(area type : area of typical (high floor) apartment)

(area feature : (beautiful and clean area))

(area feature : ((big opened window) with view to sky))

(area feature : (dimension : (2m 4m 2.5m)))

((object in area) : (myTireColumn stay in (floor and (very close to the window))))

)

(end fam language)

{This code ask AI system to generate image, which must precisely describe the scene.

Copy this code, paste this code to chat windows with AI system, then it will give the image of precise scene : an array of automobile tire stay in specific drying area}

**Example 4 :**

(begin fam language)

(set variable ((variable name) is ‘a’, (variable value) is 123))

(set variable ((variable name) is ‘b’, (variable value) = 456))

(repeat (print (a + b + index)) (5 time) (index go from 1 to 5)

(end fam language)

{ Paste the above fam language code to AI system, such as Google Gemini, Microsoft CoPilot, ChatGPT, and ask them to generate explicit programming language codes, for example Python code, Rust code, C++ code, then you will receive explicit programming code doing the intension. }

The above instruction is intension to do simple program :

+ set variable a=123

+ set variable b=456

+ repeat print (a + b + index) for 5 times, where index go from 1 to 5.

Beside , it is possible to simply write : (a = 123) to set a variable . User can choose their familiar way , instead of must find the specific way of setting variable .

This is simple program. But use meaning here is that : you do not know, for example, Python code syntax, or you know Python syntax not well, but you can describe enough precise intention in fam language, so that you receive explicit Python code to realize your intension. In natural language, it is impossible to strictly describe precise intension/instruction to AI system.

Of course, if you know very well programming languages and can very quickly remember colossal number of all detailed syntax and command of programming language, then you should not use AI, but you can do all your instructions by yourselves writing programs. But reality shows that it require big human cognitive load, and almost people are not comfortable with that. Fam language levelized, it require substantial less your cognitive load for syntax memory, while allow you to maximal strictly describe precise matter, and allow minimal consume energy of AI calculating at server side.

One of the most powerful use case of fam language : you use fam language to possible precise instruction to AI system, so that AI system will generate programming code (for example Rust code, Python code, C++ code, .. ) realizing your intension. Then you correct the received code .

## 2.10/ Fam language as universal AI-AI language :

Fam language is also perfectly universal AI-AI language to share and exchange knowledge between different AI system of different specific architecture and specific model .

Fam language specific instruction and specification for AI-nonbiological user is described in next part , which is dedicated for nonbiological user (AI user) to write fam language message to exchange knowledge with AI system . It allow more precisely describe matter in fam language. Of course it do not restrict human user to write and use these fam language specific command .

# 3/ Instruction to interpret fam language code for AI system :

Fam language code are store in many ‘hypertext’ file (fam language code page file). (For simple, usually use simple standard text file to store fam language code).

Each fam language code page contain many ordered code line , where fam language element (other name : fam language code element) are written. It is typical that fam language code page is simply text file, where each 1 fam language code element is written in 1 single line or in multi-**consecutive** line .

Possible use the term ‘fam language message’ or ‘fam language command’ for short fam language code page. For example, when human user send short fam language text to chat window to AI system , these short fam language text are short fam language code page, and are called by term ‘fam language message’ or ‘fam language command’ .

Each fam language code page have its own 1 strict hierarchy structure of this fam language code page. The fam language code page is 1 combined code element at the 0-level of its own strict hierarchy structure of the fam language code page. All element of the fam language code page are nested element, which stay in different higher level in the strict hierarchy structure of the fam language code page.

Thus when the AI system load the fam language code page (or other used term ‘fam language command’ , ‘fam language message’), the AI system receive full information (full contain) of each element of the loaded fam language code page, and the strict hierarchy structure of the loaded fam language code page, and all own hierarchy structure of all element of the loaded fam language code page. According to the ‘Definite Number Of Element Principle Of Fam Language’ (refer to part 2.4.2/) , definite number of element of fam language element of the loaded fam language code page allow the AI system can quickly and easy to define whether 2 arbitrary element from (the definite number of element of the loaded fam language code page) are identical equal. Because element of the loaded fam language code page are written in text file (or in text message , or in hypertext file , or in hypertext message), thus the writing-element-description-code of each element are simply text string . Thus if writing-element-description-code of the one element is the same as the writing-element-description-code of the other element in the current fam language code page, then the AI system must reckon that this one element and this other element are absolute identically equal.

Fam language specify the fam language specific keyword element ‘###’ to refer to 1 the primary containing element, which contain this keyword element ‘###’ at the 1-level in strict hierarchy structure of this containing element.

Interpret fam language code, or parse fam language code, or compile fam language code, all these task do not mean to directly convert fam language code into final machine (computer hardware) code.

Interpret fam language code, or parse fam language code, or compile fam language code, all these task mean that the AI system read fam language code, then the AI system generate interpretation (of this fam language code), which is optimal corresponding to the intention of this fam language code.

Because fam language is abstract language for universal use case of all general and professional user, thus the pure fam language version , which is described in this specification , try to keep as few as possible fam language specific command name and keyword and word, to minimize the required human user cognitive load of fam language specific name memorization.

All fam language specific strict instruction for interpreting , to instruct AI system to interpret fam language code into some conceptual temporary interpretation (or simply temporary interpretation) , and then AI system will continue to interpret these temporary interpretation into final interpretation of fam language code. The process , AI system follow fam language specific strict instruction for interpreting , is explicit and strict . The process , AI system interpret temporary interpretation of fam language code into final interpretation of fam language code , is depended on AI system specific architecture and model. And thus , The process , AI system interpret temporary interpretation of fam language code into final interpretation of fam language code , is the guaranteeing of universal abstract feature of fam language . In this part, when necessary , there will be note that which instruction are fam language specific strict instruction . The pure version of fam language levelized, it try to keep as less as possible fam language

specific strict instruction of interpreting , so that it require as less as possible human user brain cognitive load of fam language specific syntax and word and name memorization , while it also allow to comfortable do most all task in general. Other future version of fam language can add more language specific function and specific name for rare advanced task.

The AI system read fam language code, then the AI system apply the following fact:

* fam language specific strict instruction : especially fam language specific rule for element-replacement .

* contain of element of fam language code page, and element abstract relationship in strict hierarchy structure ;

* AI system knowledge ;

* especially AI system must exploit the coincidence between (human user standard natural language vocabular, human user standard English vocabulary) with (AI system predefined natural language vocabulary, predefined AI system English vocabulary) ;

* and AI system must exploit the coincidence between (human user standard knowledge in other sector, for example in priority , standard mathematic , standard logic , standard multi-science , standard programming language , standard culture) with (AI system predefined standard knowledge in other sector, for example in priority , standard mathematic , standard logic , standard multi-science , standard programming language , standard culture) ;

--> for the AI system guess and prognose the intention of fam language code (human user intention , which are written in fam language code) , so that AI system can generate the intended interpretation of fam language code.

## 3.1/ Fam language specific strict instruction for AI system to interpret fam language code :

Fam language specific strict instruction do 1 of 2 task : when they interpret element , they replace 1 met element by another specified element . Or they add or modify replacement rule in the fam language codepage rule repository to serve the task of replacing other met element .

Fam language specific strict instruction are rule , which show how AI system must replace 1 arbitrary element ‘AA’ of the current fam language codepage by another 1 specified element ‘BB’ in the updated fam language codepage when AI system meet this element ‘AA’ in the current fam language codepage . It mean that when AI system meet 1 arbitrary element ‘AA’ in the current fam language codepage , then the AI system must follow fam language specific rule to replace the 1 element ‘AA’ by the other 1 element ‘BB’ . Thus the AI system will update fam language codepage step-by-step , by solely fam language specific strict rule , into many intermediate fam language codepage , and finally into the final fam language codepage , which is the conceptual interpretation of the original fam language codepage. After that AI system will use their private own AI system specific architecture and model , to interpret the fam language specific conceptual interpretation of fam language codepage into final interpretation of fam language codepage . General feature of fam language rule is that they replace strictly 1 arbitrary element by 1 other arbitrary specified element , 1 by 1 , but not 1 by 2 .

All fam language specific strict instruction of interpreting fam language code are explicit instruction, which are guaranteed to be feasible realizable in all universal use case , and do not depend on AI system specific architecture and model.

To specify fam language specific strict instruction for AI system to interpret fam language code , fam language use dedicated fam language specific code pattern element . Each fam language code pattern is a set of fam language element , which satisfy some specified condition of pattern . Use the term ‘fam language code pattern element’ to specify that the fam language code pattern element are belong to some fam language code pattern . The throughout universal feature of all fam language element in general , and fam language code pattern element in particular , is that they all must strictly be written in the 1 universal syntax of fam language .

Beside , all natural integer are fam language keyword element : 0 , 1 , 2 , .. so on ..

One of the soul unique feature of fam language is that it try as much as possible to avoid using code text regular expression to define specific function and command . All fam language specific function and command are defined by fam language specific code pattern element , which are themselve fam language element in the form of 1 and only 1 robust universal syntax of fam language .

Fam language is not only the high level universal human-AI language , but also the high level universal AI-AI language . In fam language , human user can declare function , and then call function , by many of their familiar way , for example by natural language function declaration and function calling , and by popular programming language function declaration and function calling , and by popular tech sector function declaration and function calling . The AI system , which already have predefined knowledge about these popular different way of function declaration and function calling in various different sector , will quickly and consistently understand any these way of function declaration and function calling from human people . And it is the recommended way for general purpose for human user to define function and call function in fam language . But for AI-AI system , the concept of readability is different . Fam language specific code pattern element , and fam language specific interpreting instruction are INDEPENDENT AT ANY AI SYSTEM SPECIFIC ARCHITECTURE , and are dedicated tool for AI system to more precisely describe their knowledge and idea in fam language message (fam language codepage) to send AI-generated fam language message to other AI system . Thus fam language use fam language code pattern element , instead of text regular expression , to declare function and call function , for the purpose that all clearly fam language specific code pattern element in form of 1 robust universal syntax of fam language will allow all AI system to quickly and consistently parse all fam language function . It create throughout universal consistency in all fam language world between all AI system and human user about that all fam language specific function declaration and function calling are simply fam language specific code pattern element in form of 1 universal syntax of fam language . Thus all fam language specific code pattern are dedicated for AI-AI communication , but it do not require general human user to learn fam language specific code pattern element , and it also do not restrict human user to use fam language specific code pattern element . If human user write fam language specific code pattern element , then the impact and implication are the same as AI system write fam language code pattern element .

Different AI system can generate fam language message , which can use fam language code pattern element to describe any precise essence , and then to send generated fam language message to other AI system to exchange AI-AI knowledge . Because fam language code pattern element and fam language specific strict instruction are independent at any AI system architecture and model . Thus all AI system can keep their COMMERCIAL specific proprietary architecture and model in the AI – AI knowledge exchanging via fam language .

Professional AI system biological developer of course can find usefulness of using fam language code pattern element to write different fam language message to give knowledge to AI system . Professional AI system nonbiological developer , who are not biology people , can find usefulness of using fam language code pattern element to write different fam language message to give knowledge to AI system .

### 3.1.0/ Fam language specific strict instruction for interpreting : fam language specific inherent index of element :

**The inherent index of element :** Lets suppose that an element ‘AA’ is contained inside an element ‘A’ , it mean that the element ‘AA’ stay somewhere in the own hierarchy structure of the element ‘A’ . The element ‘AA’ in the own hierarchy structure of the element ‘A’ can be specified by a unique sequence of ordered integer , which is the inherent index of element ‘AA’ inside element ‘A’ .

For example :

Code : (e1 AA e2 (e3 e4))

In the above example , inherent index of the element ‘AA’ inside the element ‘(e1 AA e2 (e3 e4))’ is (1) ,

inherent index of the element ‘e3’ inside the element ‘(e1 AA e2 (e3 e4))’ is (3 0) , inherent index of the element ‘e4’ inside the element ‘(e1 AA e2 (e3 e4))’ is (3 1) .

For example :

Code : (e1 AA AA (e2 e3 e4 (AA)))

In the above example , there are 3 element ‘AA’ : first element ‘AA’ , second element ‘AA’ , third element ‘AA’ .

Inherent index of the first element ‘AA’ inside the element ‘(e1 AA AA (e2 e3 e4 (AA)))’ is (1) ,

Inherent index of the second element ‘AA’ inside the element ‘(e1 AA AA (e2 e3 e4 (AA)))’ is (2) ,

Inherent index of the third element ‘AA’ inside the element ‘(e1 AA AA (e2 e3 e4 (AA)))’ is (3 3 0) .

Thus in fam language , inherent index of element ‘AA’ inside element ‘A’ is the element , whose all 1-level nested element are integer , which specify the element ‘AA’ in the own hierarchy structure of the element ‘A’ .

Use the term ‘{n}-level containing element’ : Given arbitrary element ‘AA’ . The {n}-level containing element of the element ‘AA’ is the element , which contain the element ‘AA’ at the {n}-level of the own hierarchy structure of the {n}-containing element of this element ‘AA’ .

For example :

Code : (element1 (element2 (element3 element4)) element5)

In this example , the 1-level containing element of the element ‘element3’ is the element ‘(element3 element4)’ , the 2-level containing element of the element ‘element3’ is the element ‘(element2 (element3 element4))’ , the 3-level containing element of the element ‘element3’ is the element ‘(element1 (element2 (element3 element4)) element5)’ .

To find the inherent index of the element ‘AA’ inside the element ‘A’ , for example , the inherent index in the form (I{1} I{2} .. I{n}) , where I{1} , I{2} , .. , I{n} are integer , the AI system can find from the last integer I{n} to the first integer I{1} . Thus the integer I{n} is the 1-level index of the element ‘AA’ in the 1-level containing element of element ‘AA’ . The integer I{n-1} is the 1-level index of the element (1-level containing element of element ‘AA’) in the element (2-level containing element of the element ‘AA’) . The integer I{n-2} is the 1-level index of the element (2-level containing element of the element ‘AA’) in the element (3-level containing element of the element ‘AA’) . So on .. , I{1} is the 1-level index of the element ({n-{n-1}}-level containing element of the element ‘AA’) in the element ‘A’ , which is the {n}-level containing element of the element ‘AA’ .

The pure version of fam language specify 3 type of inherent index of element : the local inherent index of element , the region inherent index of element , and the relative universal inherent index of element .

**3.1.0.1/ Local inherent index of element :** Given an arbitrary element ‘A’ . Local inherent index of element ‘A’ is the inherent index of element ‘A’ inside the 1-level containing element of the element ‘A’ .

For example :

Code : (B C K A (F G))

In the above example , the local inherent index of the element ‘A’ is (3) , the local inherent index of the element ‘G’ is (1) , the local inherent index of the element ‘(F G)’ is (4) .

**3.1.0.2/ Region inherent index of element : {n}-deep region inherent index of element** : Given an arbitrary element ‘A’ . The {n}-deep region inherent index of the element ‘A’ is the inherent index of the element ‘A’ inside the {n}-level containing element of the element ‘A’ .

For example :

Code : (E1 E2 (B1 (B2 ((BB1 BB2) (BB3 BB4)) B3)))

In the above code , the 1-deep region inherent index of the element ‘BB2’ is (1) , the 2-deep region inherent index of the element ‘BB2’ is (0 1) , the 3-deep region inherent index of the element ‘BB2’ is (1 0 1) , the 4-deep region inherent index of the element ‘BB2’ is (2 1 0 1) .

**3.1.0.3/ Relative universal inherent index of element** : Given an arbitrary element ‘A’ , the {n}-deep region inherent index of the element ‘A’ depend on integer ‘n’ . When increase integer ‘n’ , then the {n}-level containing element of the element ‘A’ become bigger and bigger . In each use context , the integer ‘n’ can reach to some limited maximum value nMax , at this value nMax , the {nMax}-deep region inherent index of the element ‘A’ is called relative universal inherent index of the element ‘A’ . Thus at each moment the relative universal inherent index of element ‘A’ is the inherent index of the element ‘A’ in the current runtime updated fam language codepage .

For example :

Code of the fam language codepage :

Code line1 : (AA BB (CC DD))

Code line2 : (11 (22 33) 44)

In the above example , suppose that there is no space sequence between element ‘(AA BB (CC DD))’ and element ‘(11 (22 33) 44)’ , except the abstract separating space .

Thus the relative universal inherent index of element ‘22’ is the inherent index of the element ‘22’ in the fam language codepage , and is (1 1 0) .

**3.1.0.4/ Fam language specific Logical Executing Order of ordered integer sequence : compare logical executing order of ordered integer sequence .**

Ordered integer sequence is sequence of integer.

Each fam language element , whose all 1-level nested element are integer , is ordered integer sequence . Thus fam language inherent index element are ordered integer sequence too .

For example :

Code : (35 1 2 3 7 2)

In the above code , the fam language element ‘(35 1 2 3 7 2)’ is a ordered-6-integer sequence , and it can be possible fam language inherent index element .

Fam language should not specify the concrete concept of Logical Executing Order of ordered integer sequence , but fam language should explicitly specify the comparison between Logical Executing Order of one fam language specific inherent index element and Logical Executing Ordere of another fam language specific inherent index element .

(I{1} I{2} .. I{n})

(J{1} J{2} .. J{m})

Above are 2 arbitrary ordered integer sequence , where I{1} , I{2} , .. , I{n} and J{1} , J{2} , .. , J{m} are integer .

**Case 1 :** If m > n , and I{1} = J{1} , I{2} = J{2} , .. , I{n} = J{n}

In this case , say that the sequence ‘(J{1} J{2} .. J{m})’ is higher logical executing order than the sequence ‘(I{1} I{2} .. I{n})’ , or by other word : the sequence ‘(I{1} I{2} .. I{n})’ is lower logical executing order than the sequence ‘(J{1} J{2} .. J{m})’ .

For example :

I = (0 3 5)

J = (0 3 5 0)

In the above example , the sequence ‘J’ is higher logical executing order than the sequence ‘I’ .

For example :

I = (2 7 6 0)

J = (2 7 6 0 2 3)

In the above example , the sequence ‘J’ is higher logical executing order than the sequence ‘I’

**Case 2 :** If there exist an integer p , that I{1} = J{1} , I{2} = J{2} , .., I{p} = J{p} , and I{p+1} < J{p+1}

In this case , say that the sequence ‘(J{1} J{2} .. J{m})’ is higher logical executing order than the sequence ‘(I{1} I{2} .. I{n})’ , or by other word : the sequence ‘(I{1} I{2} .. I{n})’ is lower logical executing order than the sequence ‘(J{1} J{2} .. J{m})’

For example :

I = (3 5 0 1 5 9 6 55 6)

J = (3 5 0 1 7 5 3)

In this example , the sequence ‘J’ is higher logical executing order than the sequence ‘I’ , or by other word : the sequence ‘I’ is lower logical executing order than the sequence ‘J’ .

**Case 3 :** If there exist an integer p , that I{1} = J{1} , I{2} = J{2} , .., I{p} = J{p} , and I{p+1} > J{p+1}

In this case , say that the sequence ‘(J{1} J{2} .. J{m})’ is lower logical executing order than the sequence ‘(I{1} I{2} .. I{n})’ , or by other word : the sequence ‘(I{1} I{2} .. I{n})’ is higher logical executing order than the sequence ‘(J{1} J{2} .. J{m})’

For example :

I = (0 3 6 1 2 55 6)

J = (0 3 4 5 3)

In this example , the sequence ‘J’ is lower logical executing order than the sequence ‘I’ , or by other word : the sequence ‘I’ is higher logical executing order than the sequence ‘J’ .

**Case 4 :** If I{1} < J{1} ,

In this case , say that the sequence ‘(J{1} J{2} .. J{m})’ is higher logical executing order than the sequence ‘(I{1} I{2} .. I{n})’ , or by other word : the sequence ‘(I{1} I{2} .. I{n})’ is lower logical executing order than the sequence ‘(J{1} J{2} .. J{m})’

For example :

I = (8 99 8 52 4 2 1)

J = (9 0 3)

In this example , the sequence ‘J’ is higher logical executing order than the sequence ‘I’ , or by other word : the sequence ‘I’ is lower logical executing order than the sequence ‘J’ .

**Case 5 :** If I{1} > J{1} ,

In this case , say that the sequence ‘(J{1} J{2} .. J{m})’ is higher logical executing order than the sequence ‘(I{1} I{2} .. I{n})’ , or by other word : the sequence ‘(I{1} I{2} .. I{n})’ is lower logical executing order than the sequence ‘(J{1} J{2} .. J{m})’

For example :

I = (8 99 8 52 4 2 1)

J = (9 0 3)

In this case , say that the sequence ‘(J{1} J{2} .. J{m})’ is lower logical executing order than the sequence ‘(I{1} I{2} .. I{n})’ , or by other word : the sequence ‘(I{1} I{2} .. I{n})’ is higher logical executing order than the sequence ‘(J{1} J{2} .. J{m})’

For example :

I = (1 5 1 1 2 0)

J = (0 9 5 6 3 4 8 9)

In this example , the sequence ‘J’ is lower logical executing order than the sequence ‘I’ , or by other word : the sequence ‘I’ is higher logical executing order than the sequence ‘J’ .

**3.1.0.5/ Fam language specific Logical Executing Order of a fam language element inside another fam language element :**

Given an arbitrary element ‘AA’ inside the other element ‘A’ .

Fam language specific logical executing order of the element ‘AA’ in the element ‘A’ is the logical executing order of (the inherent index of the element ‘AA’ inside the element ‘A’) . Thank to this definition , now can compare the logical executing order of element ‘AA1’ in the element ‘A’ with the logical executing order of element ‘AA2’ in the element ‘A’ : by to compare the logical order of (the inherent index of the element ‘AA1’ inside the element ‘A’ ) with the logical order of (the inherent index of the element ‘AA2’ inside the element ‘A’ ) .

**Principle** : When need to compare the logical executing order of different element inside the 1 same containing element , then need to compare the logical executing order of these element inherent index in the same containing element .

Given 2 arbitrary element ‘AA1’ and ‘AA2’ inside the other element ‘A’ . Say that the element ‘AA1’ is higher logical executing order than the element ‘AA2’ inside the element ‘A’ if (the inherent index of the element ‘AA1’ in the element ‘A’) is higher logical executing order than (the inherent index of the element ‘AA2’ in the element ‘A’) . Equivalent word : the element ‘AA2’ is lower logical executing order than the element ‘AA1’ inside the element ‘A’ .

Say that the element ‘AA2’ is higher logical executing order than the element ‘AA1’ inside the element ‘A’ if (the inherent index of the element ‘AA2’ in the element ‘A’) is higher executing order than (the inherent index of the element ‘AA1’ in the element ‘A’) . Equivalent word : the element ‘AA1’ is lower logical executing order than the element ‘AA2’ inside the element ‘A’ .

For example :

Code : (A B (C A CA (BB D (E F) K)) P Q (L G H))

In the above code , lets compare logical executing order of some element inside this big element ‘(A B (C A CA (BB D (E F) K)) P Q (L G H))’ by to compare logical executing order of their inherent index in this big element ‘(A B (C A CA (BB D (E F) K)) P Q (L G H))’

The element ‘A’ have inherent index ‘(0)’ , the element ‘B’ have inherent index ‘(1)’ , and thus the element ‘B’ is higher logical executing order than the element ‘A’ inside this big element .

The element ‘BB’ have inherent index ‘(2 3 0)’ , the element ‘CA’ have inherent index ‘(2 2)’ , and thus the element ‘BB’ is higher logical executing order than the element ‘CA’ inside this big element .

The element ‘E’ have inherent index ‘(2 3 2 0)’ , the element ‘(E F)’ have inherent index ‘(2 3 2)’ .

The element ‘(BB D (E F) K)’ have inherent index ‘(2 3)’ .

The element ‘(L G H)’ have inherent index ‘(5)’ .

Thus can place the above listed element in sequence of rising logical executing order : ‘A’ , ‘B’ , ‘CA’ , ‘(BB D (E F) K)’ , ‘BB’ , ‘(E F)’ , ‘E’ , ‘(L G H)’ . Because their corresponding inherent index are ‘(0)’ , ‘(1)’ , ‘(2 2)’ , ‘(2 3)’ , ‘(2 3 0)’ , ‘(2 3 2)’ , ‘(2 3 2 0)’ , ‘(5)’ , which also stay in a sequence of rising logical execution order of ordered integer sequence .

**3.1.0.5.1/ Fam language specific region logical executing order of element : {n}-deep region logical executing order of element :**

Given an arbitrary element ‘A’ . The {n}-deep region logical executing order of the element ‘A’ is the logical executing order of the element ‘A’ inside its {n}-deep containing element .

(The {n}-deep containing element of an element ‘A’ is the element , which contain the element ‘A’ at the n-level of its own hierarchy structure)

For example :

Code : (A B (E F (G (H1 H2 H3) K)) (P Q) L)

In the above example , the 1-deep region logical executing order of the element ‘H3’ is the logical executing order of ‘(2)’ . The 2-deep region logical executing order of the element ‘H3’ is the logical executing order of ‘(1 2)’ . The 3-deep region logical executing order of the element ‘H3’ is the logical executing order of ‘(2 1 2)’ . The 4-deep region logical executing order of the element ‘H3’ is the logical executing order of ‘(2 2 1 2)’ .

**3.1.0.5.2/ Relative universal logical executing order of element :**

Relative universal logical executing order of element is the logical executing order of this element inside the current runtime updated fam language codepage element .

**3.1.0.6/ fam language specific code pattern element : local-inherent-index-of-element rule :**

**Purpose 1 : Specify local inherent index of element :**

The fam language specific code pattern element : **([{!}] 1)**

Where the element ‘[{!}]’ is fam language specific keyword element , the element ‘1’ is fam language specific keyword natural number element .

These specific code pattern element always begin with parenthese ‘(‘ and end with parenthese ‘)’ , always have strictly 2 nested element at 1-level of its own hierarchy structure : ‘[{!}]’ , ‘1’ with space between them .

The AI system interpret this specific code pattern element ‘([{!}] 1)’ by the fam language specific (local-inherent-index-of-element) rule :

When the AI system meet this fam language specific code pattern element ‘([{!}] 1)’ in the fam language codepage , then the AI system replace this element ‘([{!}] 1)’ by (the inherent index of this element ‘([{!}] 1)’ inside the 1-level containing element of this element ‘([{!}] 1)’) .

Feature : Because the inherent index of arbitrary element ‘A’ inside its 1-level containing element is always a element , which contain only 1 integer as 1-level nested element . Thus the inherent index of the element ‘([{!}] 1)’ inside its 1-level containing element is the element , which contain only 1 integer as 1-level nested element .

Feature : After the AI system replace the element ‘([{!}] 1)’ by the result element , which is the local inherent index of the element ‘([{!}] 1)’ , then the local inherent index of this result element is absolute identical equal to this result element too .

For example :

Code : (E B 12 (123 ([{!}] 1) C) ([{!}] 1) C K G B)

In the above example , the first element ‘([{!}] 1)’ have its inherent index ‘(1)’ inside its 1-level containing element ‘(123 ([{!}] 1) C)’ , the second element ‘([{!}] 1)’ have its inherent index ‘(4)’ inside its 1-level containing element ‘(E B 12 (123 ([{}] 1) C) ([{}] 1) C K G B)’ .

Interpreted result must be : (E B 12 (123 (1) C) (4) C K G B)

In this result , the local inherent index of the element ‘(1)’ is the element ‘(1)’) , and the local inherent index of the element ‘(4)’ is the element ‘(4)’ .

**Purpose 2 : use case of the fam language specific code pattern element ([{!}] 1)**

Fam language do not use text regular expression to declare function , but it prefer to use fam language code pattern element in its 1 universal syntax , to declare function . Thus , in fam language codepage , it is impossible to directly explicitly write function to get the inherent index of the element ‘A’ without changing the structure of the 1-level containing element of the element ‘A’ .

Possible use the fam language specific code pattern element ‘([{!}] 1)’ to infer some information about its neighbor element .

For example :

Code : (E B 12 (123 ([{!}] 1) C) ([{!}] 1) C K G B)

The AI system use fam language specific strict instruction to interpret this code , the result must be :

(E B 12 (123 (1) C) (4) C K G B)

In the result element ‘(E B 12 (123 (1) C) (4) C K G B)’ , note about the position of the 2 element ‘C’ . The first element ‘C’ stay next right to the element ‘(1)’ , which is the local inherent index of the element ‘(1)’ too . The second element ‘C’ stay next right to the element ‘(4)’ , which is the local inherent index of the element ‘(4)’ too . Thus the local inherent index of the first element ‘C’ is ‘(2)’ , and the local inherent index of the second element ‘C’ is ‘(5)’ .

**3.1.0.7/ Fam language specific code pattern : {n}-region-inherent-index-of-element rule :**

**Purpose 1 : Specify the {n}-deep region inherent index of element .**

The fam language specific code pattern element : **([{!}] N)**

Where the element ‘[{!}]’ is fam language specific keyword element ; element ‘N’ is arbitrary element .

These specific code pattern element always begin with parenthese ‘(‘ and end with parenthese ‘)’ , always have strictly 2 nested element at 1-level of its own hierarchy structure : ‘[{!}]’ , ‘N’ with space between them .

The AI system interpret this specific code pattern element ‘([{!}] N)’ by the fam language specific ({n}-deep-region-inherent-index-of-element) rule :

When the AI system meet this fam language specific code pattern element ‘([{!}] N)’ in the fam language codepage , then the AI system firstly update the element ‘N’ by using other fam language specific replacement rule , as a result : the element ‘N’ -> the element ‘updatedN’.

If the element ‘updatedN’ is not integer , then the AI system replace the element ‘([{!}] N)’ by the element ‘()’ .

If the element ‘updatedN’ is integer ‘0’ , then the AI system replace the element ‘([{!}] N)’ by the element ‘()’ .

If the element ‘updatedN’ is integer greater 0 , then the AI system replace the element ‘([{!}] N)’ by (the {updatedN}-deep region inherent index of the element ‘([{!}] N)’) . But if the AI system can not find (the {updatedN}-deep region inherent index of the element ‘([{!}] N)’) because the AI system can not locate the ({updatedN}-level containing element of the element ‘([{!}] N)’) , for example because out of index range , then the AI system replace the element ‘([{!}] N)’ by the element ‘()’ .

Feature : When the AI system replace the element ‘([{!}] N)’ by (its {updatedN}-deep region inherent index) , which is not ‘()’ , then this result element contain only {updatedN} 1-level nested element , which are integer .

Feature : When the AI system replace the element ‘([{!}] N)’ by its {updatedN}-deep region inherent index , which is not ‘()’ , then this result element and (its {updatedN}-deep region inherent index) are absolute identical equal .

For example :

Code : (E B 12 (123 ([{!}] 2) C) ([{!}] 1) C K G B)

In the above example , the element ‘([{!}] 2)’ have its 2-deep region inherent index ‘(3 1)’ . The element ‘([{!}] 1)’ have its local inherent index ‘(4)’ .

Interpreted result must be : (E B 12 (123 (3 1) C) (4) C K G B)

In this result , the 2-deep region inherent index of the element ‘(3 1)’ is the element ‘(3 1)’) , and the local inherent index of the element ‘(4)’ is the element ‘(4)’ .

**Purpose 2 : use case of the fam language specific code pattern element ([{!}] N)**

Fam language do not use text regular expression to declare function , but it prefer to use fam language code pattern element in its 1 universal syntax , to declare function . Thus , in fam language codepage , it is impossible to directly explicitly write function to get the {n}-deep region inherent index of the element ‘A’ without changing the structure of the 1-level containing element of the element ‘A’ .

Possible use the fam language specific code pattern element ‘([{!}] N)’ to infer some information about its neighbor element .

For example :

Code : (E B 12 (123 ([{!}] 2) C) ([{!}] 1) C K G B)

The AI system use fam language specific strict instruction to interpret this code , the result must be :

(E B 12 (123 (3 1) C) (4) C K G B)

In the result element '(E B 12 (123 (3 1) C) (4) C K G B)' , note about the position of the 2 element 'C' . The first element 'C' stay next right to the element '(3 1)' , which is the 2-deep region inherent index of the element '(3 1)' too . The second element 'C' stay next right to the element '(4)' , which is the local inherent index of the element '(4)' too . Thus the 2-deep region inherent index of the first element 'C' is '(3 2)' , and the local inherent index of the second element 'C' is '(5)' .

**3.1.0.8/ Fam language specific code pattern : relative-universal-inherent-index-of-element rule :**

**Purpose 1 : Specify relative-universal-inherent-index of element :**

The fam language specific code pattern element : **([{!}] […])**

Where the element '[{!}]' is fam language specific keyword ; the element '[…]' is fam language specific keyword .

These specific code pattern element always begin with parenthese '(' and end with parenthese ')' , always have strictly 2 nested element at 1-level of its own hierarchy structure : '[{!}]' , '[…]' with space between them .

The AI system interpret this specific code pattern element '([{!}] […])' by the fam language specific (relative-universal-inherent-index-of-element) rule :

When the AI system meet this fam language specific code pattern element '([{!}] […])' in the fam language codepage , then the AI system replace this element '([{!}] […])' by the inherent index of this element '([{!}] […])' inside the current updated fam language codepage element .

Feature : In each updated fam language codepage element , each element have its unique relative universal inherent index .

Feature : when AI system add small fam language codepage element (fam language message) to 1 big another fam language codepage element of a chat session , then the relative universal inherent index of all element , which are contained in this old small fam language codepage (fam language message) , will also be changed to the corresponding new fam language codepage of chat session.

For example :

Code in the fam language message :

(something1 something2 (E B 12 (123 ([{!}] […]) C) ([{!}] […]) C K G B) something3)

In the above example , the first element '([{!}] […])' have its relative universal inherent index '(2 3 1)' , the second element '([{}] 1)' have its relative universal inherent index '(2 4)' .

Interpreting result must be :

(something1 something2 (E B 12 (123 (2 3 1) C) (2 4) C K G B) something3)

**Purpose 2 : use case of the fam language specific code pattern element ([{!}] […])**

Fam language do not use text regular expression to declare function , but it prefer to use fam language code pattern element in its 1 universal syntax , to declare function . Thus , in fam language codepage , it is impossible to directly explicitly write function to get the inherent index of the element ‘A’ without changing the structure of the 1-deep containing element of the element ‘A’ .

Possible use the fam language specific code pattern element ‘([{!}] […])’ to infer some information about its neighbor element .

For example :

Code : Code in the fam language message :

(something1 something2 (E B 12 (123 ([{!}] […]) C) ([{!}] […]) C K G B) something3)

The AI system use fam language specific strict instruction to interpret this code , the result must be :

(something1 something2 (E B 12 (123 (2 3 1) C) (2 4) C K G B) something3)

In the result , note about the position of the 2 element ‘C’ . The first element ‘C’ stay next right to the element ‘(2 3 1)’ , which is the relative universal inherent index of the element ‘(2 3 1)’ too . The second element ‘C’ stay next right to the element ‘(2 4)’ , which is the relative universal inherent index of the element ‘(2 4)’ too . Thus the relative universal inherent index of the first element ‘C’ is ‘(2 3 2)’ , and the relative universal inherent index of the second element ‘C’ is ‘(2 5)’ .

**3.1.0.9/ Theorem about the relation between inherent index of the containing element with inherent index of its nested element :**

Given an arbitrary containing element ‘A’ , which contain other nested element .

Call the IA = (I{1} I{2} .. I{n}) as the relative universal inherent index of the element ‘A’ , where I{1} , I{2} , .. , I{n} are integer .

The following theorem is true : all contained element of the element ‘A’ must have relative universal inherent index (I{1} I{2} .. I{n} I{n+1} I{n+2} .. I{m}) . It mean that first {n} nested element of the relative universal inherent index of the contained element of the element ‘A’ must be I{1} , I{2} , .. , I{n} .

If IA is not the relative universal inherent index of the element ‘A’ , but is the inherent index of the element ‘A’ in specified element ‘BigElement’ , then there is analogous confirmation : all contained element of the element ‘A’ must have their inherent index inside the element ‘BigElement’ as (I{1} I{2} .. I{n} I{n+1} I{n+2} .. I{m}) . It mean that first {n} nested element of the inherent index of the contained element of the element ‘A’ must be I{1} , I{2} , .. , I{n} .

Thus the AI system can use the relative universal inherent index of element to fast check the whether an element is contained inside another element .

**3.1.0.10/ Theorem about invariant of inherent index of element when apply fam language specific strict instruction interpreting rule (fam language specific replacement rule) :**

When the fam language codepage are continuously updated by only the fam language specific replacement rule , then the inherent index of an arbitrary element ‘A’ in updated fam language codepage is unchanged .

Because the fam language specific strict instruction interpreting rule strictly keep the principle : replace only 1 in-line met element by 1 element , always 1 by 1 , but not 1 by 2 . Thus the fam language specific strict instruction interpreting rule can annihilate element ‘A’ (make element ‘A’ to be disappeared in fam language code page) when this element ‘A’ is contained inside a replaced element , but fam language strict instruction interpreting rule do not change inherent index of element ‘A’ when the element ‘A’ do not disappear .

This theorem allow the AI system to more effectively manage and update inherent index of all element when update fam language codepage and implement fam language strict instruction interpreting rule .

**3.1.0.11/ fam language specific conceptual logical executing order of element in fam language codepage :**

The fam language specific conceptual logical executing order of all element in fam language codepage is SERIAL , by to execute (replacement - interpret) element with lower fam language specific logic executing order firstly , then to execute (replacement – interpret) element with higher fam language specific logic executing order .

The fam language specific conceptual logical executing order of all element in the fam language codepage is applied only in the first stage of interpreting when apply only fam language specific replacement rule . After that stage , the original fam language codepage already evolve into concrete conceptual fam language codepage , which is concrete intermediate conceptual interpretation of the original fam language codepage . After that , the AI system will use AI system specific knowledge to interpret this intermediate conceptual interpretation of original fam language codepage into final interpretation of fam language codepage .

For example :

Code : (E1 E2 (E3 (E4 E5 E1) E2))

In the above example , the fam language specific logic executing order of element are as follow :

‘(E1 E2 (E3 (E4 E5 E1) E2))’ , ‘E1’ , ‘E2’ , ‘(E3 (E4 E5 E1) E2)’ , ‘E3’ , ‘(E4 E5 E1)’ , ‘E4’ , ‘E5’ , ‘E1’ , ‘E2’ .

## 3.1.1/ Fam language specific strict instruction for interpreting : fam language specific absolute-identical-equality interpreting rule :

When AI system load the fam language code page (or other used term ‘fam language command’ , ‘fam language message’), the AI system receive full information (full contain) of each element of the loaded fam language code page, and the strict hierarchy structure of the loaded fam language code page, and all own hierarchy structure of all element of the loaded fam language code page.

Among all element of the loaded fam language code page, there are some special type of element, which are called fam language specific code pattern element, or shortly specific code pattern element. Fam

language specific code pattern element are also written in the universal syntax of fam language, as any other fam language element, too. Fam language specific code pattern element are used to describe some fam language specific interpreting rule for AI system.

According to the 'Definite Number Of Element Principle Of Fam Language' (refer to part 2.4.2/) , the loaded fam language codepage can be updated into new updated fam language codepage (temporary interpretation of the original loaded fam language codepage), but at any moment , the number of all element of the updated fam language codepage (the temporary interpretation of the original loaded fam language codepage) are always different finite number .

Fam language automatically apply the code format normalization : It will eliminate all space element , which stay before the parenthese ')' and without any element between . It will eliminate all space element , which are stay after the parenthese '(' and without any element between .

For example : ( A B ( C D E ) ) -> (A B (C D E))

Because element of the fam language code page are written in text file (or in text message in the process of calculating), thus the writing-element-description-code of each element are simply text string (or hypertext string) . Thus if writing-element-description-code of the one element is the same as the writing-element-description-code of the other element in the updated fam language codepage, then the AI system must reckon that this one element and this other element are *fam-specific absolute identical equal* , or shortly *absolute identical equal* .

**The fam language specific absolute-identical-equality rule** : For 2 arbitrary fam language element , if writing-element-description-code of the first one element is the same as writing-element-description-code of the second element , then AI system must with-highest-priority reckon that these 2 element are *fam language specific absolute identical equal* , or shortly absolute identical equal , in the scope of the updated fam language codepage .

Absolute identical equality is fam language specific concept , which will used dedicatedly for fam language specific strict instruction . Absolute identical equality between 2 element do not confirm anything about the AI system concept of equality between these 2 element .

## 3.1.2/ Fam language specific strict instruction for interpreting : fam language specific (universal-element-replacement) interpreting rule .

When add this rule to fam language codepage specific rule repository , then the AI system will use this rule to replace arbitrary met 1 element by arbitrary specified 1 element .

### 3.1.2.1/ Fam language specific (universal-element-replacement-(rule declaring)) rule :

Precaution : It is not an element assignmen operation , also not temporary element storing .. It is simply a rule to directly replace element by other specified element in fam language codepage .

The fam language specific code pattern : **(Element1 [=] Element2)**

Where element 'Element1' , element 'Element2' are arbitrary element ; element '[=]' is fam language specific keyword element .

These specific code pattern element '(Element1 [=] Element2)' always begin with parenthese '(' and end with parenthese ')' , always have strictly 3 nested element at 1-level of its own hierarchy structure : 'Element1' , '[=]' , 'Element2' , with space between them .

Now given the element ‘Element1’ and given the element ‘Element2’ .

The fam language specific code pattern element : **(Element1 [=] Element2)**

Now describe the fam language specific strict instruction for AI system to interpret the fam language specific code pattern element ‘(Element1 [=] Element2)’ .

When the AI system meet such fam language code pattern element , then the AI system do not update the element ‘Element1’ and ‘Element2’ by applying other fam language specific replacement rule . Simply keep the element ‘(Element1 [=] Element2)’ , and its nested element ‘Element1’ , ‘Element2’ unchanged .

First , the AI system define the local inherent index of the element ‘(Element1 [=] Element2)’ , to later define the rule applying scope of this rule . If necessary , than the AI system can decide to find the relative universal inherent index of this rule-declaring element ‘(Element1 [=] Element2)’ to use as basic argument to define and record the rule applying scope of this rule .

Rule declaring event inherent index : it is the inherent index of the fam language specific code pattern rule-declaring element . Thus , there are : rule declaring event local inherent index , rule declaring event {n}-deep region inherent index , rule declaring event relative universal inherent index . In this case of the universal element replacement rule , they are local inherent index of the element ‘(Element1 [=] Element2)’ , {n}-deep region inherent index of the element ‘(Element1 [=] Element2)’ , relative universal inherent index of the element ‘(Element1 [=] Element2)’ . The most important are the rule declaring event local inherent index , and the rule declaring event relative universal inherent index . The AI system can also save few rule declaring event inherent index together with rule to better specify rule applying scope .

The replacement condition of rule : they are condition , when the AI system interpret some the element ‘TheElement’ , if the element ‘TheElement’ satisfy specified these condition , by which the AI system will replace the element ‘TheElement’ by other specified element .

When the AI system meet the fam language code pattern element ‘(Element1 [=] Element2)’ , then the AI system will add a new rule (replace ‘Element1’ by ‘Element2) together with the rule applying scope to the fam language codepage rule repository . Preliminary rule description : The rule (replace ‘Element1’ by ‘Element2’) mean that from this moment when the AI system interpret any other element ‘TheElement’ , which is absolute identical equal to the element ‘Element1’ , then the AI system must replace this element ‘TheElement’ by the element ‘Element2’ . The replacement condition of this rule is that : the input element must be absolute identical equal to the element ‘Element1’ . Detailed rule description : will be on next part . Rule applying scope : this rule is applied only for element , which are in this rule applying scope of this rule . The rule applying scope of this rule is the set of all element , which are contained in the 1-level containing element of the element ‘(Element1 [=] Element2)’ and are higher logical executing order than the element ‘(Element1 [=] Element2)’ . The AI system can apply 2 theorem (3.1.0.9/) and theorem (3.1.0.10/) to use the relative universal inherent index of element to check whether interested element is in the rule applying scope of this rule . The AI system can also use 2 theorem (3.1.0.9/) and theorem (3.1.0.10/) , and use the inherent index of the element ‘(Element1 [=] Element2)’ to specify and record the rule applying scope of this rule in the fam language codepage rule repository .

**3.1.2.2/ Fam language specific (universal-element-replacement) rule :**

If there were added fam language specific universal element replacement rule in the fam language codepage repository , then when AI system apply fam language specific strict instruction to interpret

element , the AI system must check whether the interested element is in the rule applying scope of specific universal element replacement rule , to replace the interested element by corresponding

Suppose that the AI system already added specific universal element replacement rule and together with rule applying scope in the fam language codepage repository before by interpreting the fam language specific code pattern (rule-declaring) element :  (Element1 [=] Element2)

**Precaution :** It is not an element assignmen operation , also not temporary element storing .. It is simply a rule to directly replace element by other specified element in fam language codepage .

When AI system meet the element ‘TheElement’ , which is absolute identical equal to the element ‘Element1’ , and is in the rule applying scope of this rule (replace ‘Element1’ by ‘Element2’) , then the AI system will do :

Case 1 : if the element ‘TheElement’ is fam language specific keyword element , then this rule will not replace the element ‘TheElement’ .

Case 2 : if the element ‘TheElement’ is predefined fam language specific code pattern element , then this rule will not replace the element ‘TheElement’ .

Case 3 : if the element ‘TheElement’ is the first nested element or the third nested element of the fam language specific code pattern rule-declaring element ‘(SomeElement [=] OtherElement)’ , where ‘SomeElement’ and ‘OtherElement’ are arbitrary element , then this rule will not replace the element ‘TheElement’ . If the element ‘TheElement’ is the 1-level nested element of the fam language code pattern rule-declaring element ‘(([] FunctionName Variable1 Variable2 .. VariableN)) [:=] G)’ , then this rule will not replace the element ‘TheElement’ .

Case 4 : If the element ‘TheElement’ is in rule applying scope of (saved universal element replacement rule1) and also in rule applying scope of (saved universal element replacement rule 2) , .. , and also in rule applying scope of (saved universal element replacement rule {n}) . And the element ‘TheElement’ satisfy all replacement condition of all these rule .

In this case , among these universal element replacement rule , the AI system will apply only 1 saved universal element replacement rule , whose rule declaring event relative universal inherent index is highest logical executing order , to the element ‘TheElement’ .

Case4 : in all other case , except Case 1 , Case 2 , Case 3 , Case 4 , then the AI system will replace the element ‘TheElement’ by the element ‘Element2’ .

For example :

Code :  (A [=] 33) (AA A (B A))

Result must be: (A [=] 33) (AA 33 (B 33))

For example :

Code : (A [=] 1) (B (A)) (A [=] 2) (A + 2)

Result must be : (A [=] 1) (B (1)) (A [=] 2) (2 + 2)

For example :

Code : (B [=] KL) (A [=] (B + C)) (B A)

Result must be : (B [=] KL) (A [=] (B + C)) (KL (B + C))

For example :

Code : (A [=] 2) ((A + 3) [=] (B + C)) (A (A + 3))

Result must be : (A [=] 2) ((A + 3) [=] (B + C)) (2 (B + C))

For example :

Code : ((A [=] 1) B A) A

Result must be : ((A [=] 1) B 1) A

For example :

Code : (A [=] KK) A (A [=] LL) A

Result must be : (A [=] KK) KK (A [=] LL) LL

Code : (A [=] PP) A ((A [=] QQ) A) A

Result must be : (A [=] PP) PP ((A [=] QQ) QQ) PP

**3.1.2.3/ Fam language specific (universal-element-replacement-(rule declaring)) rule :**

**Precaution :** It is not an element assignmen operation , also not temporary element storing .. It is simply a rule to directly replace element by other specified element in fam language codepage .

The fam language specific code pattern : **([=] Element1 Element2)**

Where element '[=]' is fam language specific keyword element ; the element 'Element1', element 'Element2' are arbitrary element .

This fam language specific code pattern rule-declaring element '([=] Element1 Element2)' is equivalent to the fam language specific code pattern rule-declaring element '(Element1 [=] Element2)' , and is imply the additional way to write .

**3.1.2.4/ Fam language specific (universal-element-replacement-in-scope (rule declaring)) rule :**

**Precaution :** It is not an element assignment operation , also not temporary element storing .. It is simply a rule to directly replace element by other specified element in fam language codepage .

The fam language specific code pattern : **([=] Element1 Element2 (M N))**

Where element '[=]' is fam language specific keyword element ; 'Element1' , 'Element2' are arbitrary element ; element 'M' and element 'N' are arbitrary element .

These specific code pattern element '(Element1 [=] Element2)' always begin with parenthese '(' and end with parenthese ')' , always have strictly 4 nested element at 1-level of its own hierarchy structure : '[=]' , 'Element1' , 'Element2' , '(M N)' with space between them . The nested element '(M N)' always begin with parenthese '(' and end with parenthese ')' , and have strictly 2 1-level nested element 'M' and 'N' with space between them .

Now given the element ‘Element1’ and element ‘Element2’ and element ‘M’ and element ‘N’ .

The fam language specific code pattern element : **([=] Element1 Element2 (M N))**

Now describe the fam language specific strict instruction for AI system to interpret the fam language specific code pattern element ‘(Element1 [=] Element2)’ .

When the AI system meet such fam language code pattern element , then the AI system do not update the element ‘Element1’ and ‘Element2’ by applying other fam language specific replacement rule . But the AI system update the element ‘M’ and element ‘N’ by applying other fam language replacement rule , as a result : the element ‘M’ -> the element ‘updatedM’ , and the element ‘N’ -> the element ‘updatedN’ .

If the element ‘updatedM’ is not natural integer greater than 0 , or the element ‘updatedN’ is not natural integer greater than 0 , then the AI system replace the element ‘([=] Element1 Element2 (M N))’ by the element ‘()’ .

If the element ‘updatedM’ is natural integer greater than 0 , and the element ‘updatedN’ is natural integer greater than 0 , and the AI system can not define the {updatedM}-level containing element of the element ‘([=] Element1 Element2 (M N))’ , then the AI system take the updated fam language codepage element as the temporary {updatedM}-level containing element of the element ‘([=] Element1 Element2 (M N))’ only for this case .

If the element ‘updatedM’ is natural integer greater than 0 , and the element ‘updatedN’ is natural integer greater than 0 , and the AI system can define the {updatedM}-level containing element of the element ‘([=] Element1 Element2 (M N))’ , then the AI replace the element ‘([=] Element1 Element2 (M N))’ by the element ‘([=] Element1 Element2 (updatedM updatedN))’ , and the AI system add a new rule (replace ‘Element1’ by ‘Element2’) together with rule applying scope , which is the set of all contained element of the {updatedM}-level containing element of the element ‘([=] Element1 Element2 (M N))’ and in the level from 1 to {updatedM + updatedN} (or to the maximal level) of the own hierarchy structure of the {updatedM}-level containing element of the element ‘([=] Element1 Element2 (M N))’ .

Rule declaring event inherent index of this rule is the inherent index of the element ‘([=] Element1 Element2 (M N))’ . Thus there are rule declaring event local inherent index of this rule , rule declaring event {n}-deep region inherent index of this rule , rule declaring event relative universal inherent index of this rule .

The fam language code pattern element **‘(Element1 [=] Element2 (M N))’** is equivalent to the fam language code pattern element ‘([=] Element1 Element2 (M N))’ , and is additional way of writing

For example :

Code : ((1 (A [=] 2) A) A (1 (A [=] 3 (1 3))) A)

Result must be : ((1 (A [=] 2) 2) A (1 (A [=] 3 (2 3))) 3)

### 3.1.3/ Fam language specific strict instruction for interpreting : fam language specific (universal-element-instant-replacement) interpreting rule .

**Precaution :** It is not an element assignment operation , also not temporary element storing .. It is simply a rule to directly replace element by other specified element in fam language codepage .

This rule is analogous to the fam language specific universal-element-replacement rule . The only difference is that , before add new element-replacement rule to the fam language codepage rule repository , the AI system firstly update the left side element (the third 1-level nested element) inside the rule-declaring element , and only after that the AI system will add new element-replacement rule to the fam language codepage rule repository .

The fam language code pattern rule-declaring element : **(Element1 [=:] Element2)**

When the AI system meet such fam language code pattern rule-declaring element , the AI system firstly update the left hand side element 'Element2' by applying other fam language replacement rule , as a result : the element 'Element2' -> 'updatedElement2' . After that , the AI system add new element-replacement rule , which is corresponding to the rule-declaring element '(Element1 [=] updatedElement2)' , in the same aforementioned described procedure .

For example :

Code : (B [=] 2) B (A [=:] (B +1)) A

Result must be : (B [=] 2) 2 (A [=:] (2 +1)) (2+1)

For example :

Code : ((B [=] 3) B (A [=:] (2 B)) A (B [=] 5) B A)

Result must be : ((B [=] 3) 3 (A [=:] (2 3)) (2 3) (B [=] 5) 5 (2 3))

**3.1.3.1/ fam language specific (universal-element-instant-replacement-in-scope) interpreting rule .**

**Precaution :** It is not an element assignment operation , also not temporary element storing .. It is simply a rule to directly replace element by other specified element in fam language codepage .

Analogous :

The fam language code pattern rule-declaring element : **(Element1 [=:] Element2 (M N))**

The fam language code pattern rule-declaring element : **([=:] Element1 Element2 (M N))** , is equivalent element , and is additional way of writing

For example :

Code : ((B [=:] C) B (A [=:] (B + C)) A (1 (A [=:] (C + B)) A 3 (A [=:] (B + 1) (2 3)) A) A)

The result must be : ((B [=:] C) C (A [=:] (C + C)) (C + C) (1 (A [=:] (C + C)) (C + C) 3 (A [=:] (C + 1) (2 3)) (C + 1)) (C + 1))

## 3.1.4 / Fam language specific strict instruction for interpreting : fam language specific (inline replacement) rule :

The fam language specific code pattern : **(AA [|] XX [->] YY)**

Where the element 'AA' and the element 'XX' and the element 'YY' are arbitrary element ; the element '[|]' and the element [->] are fam language keyword element .

These specific code pattern element always begin with parenthese '(' and end with parenthese ')' , always have strictly 5 nested element at 1-level of its own hierarchy structure , with space between them .

Given the element ‘AA’ and element ‘XX’ and element ‘YY’ .

The fam language specific code pattern element : **(AA [|] XX [->] YY)**

The AI system interpret this fam language specific code pattern element ‘(AA [|] XX [->] YY)’ by the fam language specific (inline replacement) rule :

When the AI system meet the such fam language code pattern element ‘(AA [|] XX [->] YY)’ in the current fam language codepage , then the AI system firstly not update the element ‘AA’ , but keep update and the element ‘XX’ and the element ‘YY’ by applying other fam language specific rule (fam language specific replacement rule) to update : the element ‘XX’ -> the element ‘updatedXX’ , the element ‘YY’ -> the element ‘updatedYY’ . Then the AI system replace all element inside the element ‘AA’ , which are absolute identical equal to the element ‘updatedXX’ , by the element ‘updatedYY’ , thus after all these replacement , this process already convert the element ‘AA’ into the element ‘updatedAA’ . Then the AI system replace the element ‘(AA [|] XX [->] YY)’ by the element ‘updatedAA’ . Thus the AI system update the fam language codepage .

If the element ‘updatedX’ is fam language specific keyword element , then this rule will not replace the element ‘updatedX’ .

If the element ‘updatedX’ if fam language specific code pattern element , then this rule will not replace the element ‘updatedX’ .

If the element ‘updatedX’ is the first element in the such fam language code pattern element ‘(Element1 [=] Element2) , then this rule will not replace the element ‘updatedX’ .

Thus the AI system update the fam language codepage .

This rule do not add new rule to the fam language codepage rule repository . This rule only replace the current element .

For example :

Code : ((A + B (13 C + D) (1 (1 A)) (A (A (C A)))) [|] A [->] 555)

Result must be : (555 + B (13 C + D) (1 (1 555)) (555 (555 (C 555))))

## 3.1.5 / Fam language specific strict instruction for interpreting : fam language specific (function-replacement) rule :

**Precaution :** It is not an element assignment operation , also not temporary element storing .. It is simply a rule to directly replace element by other specified element in fam language codepage .

Fam language specific instruction allow to declare function , and after declaring function then possible to call function .

There are fam language specific code pattern function-replacement function-declaring element , which are used to add new rule to the fam language codepage rule repository .

There are fam language specific code pattern function-replacement element , which are used to call function in the fam language codepage , but do not add new rule to the fam language codepage rule repository .

### 3.1.5.1/ Fam language specific function-replacement function-declaring rule :

**Precaution :** It is not an element assignment operation , also not temporary element storing .. It is simply a rule to directly replace element by other specified element in fam language codepage .

The fam language specific code pattern :

(**([] FunctionName Variable1 Variable2 .. VariableN) [:=] G)**

Where the element '[]' , '[:=]' are the fam language keyword element ; the element 'FunctionName' , 'Variable1' , 'Variable2' , .. , 'VariableN' are arbitrary element ; the element 'G' is arbitrary element .

These specific code pattern element always begin with parenthese '(' and end with parenthese ')' , always have strictly 3 nested element at 1-level of its own hierarchy structure , with space between them , :

'([] FunctionName Variable1 Variable2 .. VariableN)' , '[:=]' , 'G' .

The first 1-level nested element '([] FunctionName Variable1 Variable2 .. VariableN)' must begin with parenthese '(' and end with parenthese ')' , must have the first 1-level nested element '[]' , must have second 1-level nested element .

Given the element 'FunctionName' and element 'Variable1' and element 'Variable2' , .. , 'VariableN' .

The fam language specific code pattern function-declaring element :

(**([] FunctionName Variable1 Variable2 .. VariableN) [:=] G)**

The fam language specific code pattern function-declaring element is a fam language specific code pattern rule-declaring element .

Now describe the fam language specific strict instruction for AI system to interpret the fam language specific code pattern element '(**([] FunctionName Variable1 Variable2 .. VariableN) [:=] G)**' .

When the AI system meet such fam language code pattern element , then the AI system do not update any element inside the element '(([] FunctionName Variable1 Variable2 .. VariableN) [:=] G)' by applying other fam language specific replacement rule .

First , the AI system fine the local inherent index of the element '(([] FunctionName Variable1 Variable2 .. VariableN) [:=] G)' , to later define the rule applying scope of this rule . If necessary , than the AI system can decide to find the relative universal inherent index of this rule-declaring element '(([] FunctionName Variable1 Variable2 .. VariableN) [:=] G)' to use as basic argument to define and record the rule applying scope of this rule .

Rule declaring event inherent index : it is the inherent index of the fam language specific code pattern rule-declaring element . Thus , there are : rule declaring event local inherent index , rule declaring event {n}-deep region inherent index , rule declaring event relative universal inherent index . In this case , they are local inherent index of the element '(([] FunctionName Variable1 Variable2 .. VariableN) [:=] G)', {n}-deep region inherent index of the element '(([] FunctionName Variable1 Variable2 .. VariableN) [:=] G)' , relative universal inherent index of the element '(([] FunctionName Variable1 Variable2 .. VariableN) [:=] G)' . The most important are the rule declaring event local inherent index , and the rule declaring event relative universal inherent index . The AI system can also save few rule declaring event inherent index together with rule to better specify rule applying scope .

The replacement condition of rule : they are condition , when the AI system interpret some the element ‘TheElement’ , if the element ‘TheElement’ satisfy specified these condition , by which the AI system will replace the element ‘TheElement’ by other specified element .

When the AI system meet the fam language code pattern element ‘(([] FunctionName Variable1 Variable2 .. VariableN) [:=] G)’ , then the AI system will add a new rule (multi-replacement) together with the rule applying scope to the fam language codepage rule repository . Preliminary rule description : The rule (multi-replacement) mean that from this moment when the AI system interpret any other fam language specific code pattern element ‘([] FName Value1 Value2 .. ValueN)’ , whose first 1-level nested element is ‘[]’ , and whose second 1-level nested element ‘FName’ is absolute identical equal to the element ‘FunctionName’ , then the AI system must replace this element ‘([] FName Value1 Value2 .. ValueN)’ by the element ‘G’ , and immediate after that the AI system must apply consequently all {N} in-line replacement (Variable1 -> Value1 , Variable2 -> Value2 , .. , VariableN -> ValueN) for all element inside the element G , and as a result the AI system replace the original fam language code pattern element ‘([] FName Value1 Value2 .. ValueN)’ by the element ‘updatedG’ .

The replacement condition of this rule is that : the input element must be fam language specific code pattern element ‘([] FName Value1 Value2 .. ValueN)’ , whose first 1-level nested element ‘[]’ , and whose second 1-level nested element ‘FName’ is absolute identical equal to the element ‘FunctionName’ . Detailed rule description : will be on next part . Rule applying scope : this rule is applied only for element , which are in this rule applying scope of this rule . The rule applying scope of this rule is the set of all element , which are contained in the 1-level containing element of the function-declaring element ‘(([] FunctionName Variable1 Variable2 .. VariableN) [:=] G)’ and are higher logical executing order than the element ‘(([] FunctionName Variable1 Variable2 .. VariableN) [:=] G)’ . The AI system can apply 2 theorem (3.1.0.9/) and theorem (3.1.0.10/) to use the relative universal inherent index of the arbitrary interested element to check whether the interested element is in the rule applying scope of this rule . The AI system can also use 2 theorem (3.1.0.9/) and theorem (3.1.0.10/) , and use the rule declaring event inherent index to specify and record the rule applying scope of this rule in the fam language codepage rule repository .

**3.1.5.2/ Fam language specific (function-replacement function-calling) rule :**

**Precaution :** It is not an element assignment operation , also not temporary element storing .. It is simply a rule to directly replace element by other specified element in fam language codepage .

If there were added fam language specific function-replacement rule in the fam language codepage repository , then when AI system meet fam language specific code pattern function-calling element ‘([] FName Value1 Value2 .. ValueN)’ , the AI system must fam language function-replacement rule (which is multi-replacement rule) to replace the met fam language specific code pattern function-calling element ‘([] FName Value1 Value2 .. ValueN)’ by the specified element .

Suppose that the AI system already added fam language specific function-replacement rule and together with rule applying scope in the fam language codepage rule repository before by interpreting the fam language specific code pattern (rule-declaring) element ‘(([] FunctionName Variable1 Variable2 .. VariableN) [:=] G)’ .

When AI system meet the any fam language specific code pattern element ‘([] FName Value1 Value2 .. ValueN)’ , then the AI system will do :

The AI system will not update the 1-level nested element ‘FName’ . If the AI system see that the 1-level nested element ‘FName’ is absolute identical equal to the element ‘FunctionName’ , then the AI system will do :

The AI system will update all element ‘Value1’ , ‘Value2’ , .. , ‘ValueN’ by other fam language specific replacement rule , as a result : the element ‘Value1’ -> the element ‘updatedValue1’ , the element ‘Value2’ -> the element ‘updatedValue2’ , .. , the element ‘ValueN’ -> the element ‘updatedValueN’ .

Case 1 : if the element ‘FName’ is fam language specific keyword element , then this rule will not replace the element ‘([] FName Value1 Value2 .. ValueN)’ .

Case 2 : if the element ‘FName’ is predefined fam language specific code pattern element , then this rule will not replace the element ‘([] FName Value1 Value2 .. ValueN)’ .

Case 3 : if the element ‘TheElement’ is the first nested element or the third nested element of the fam language specific code pattern rule-declaring element ‘(SomeElement [=] OtherElement)’ , where ‘SomeElement’ and ‘OtherElement’ are arbitrary element , then this rule will not replace the element ‘TheElement’ . If the element ‘TheElement’ is the 1-level nested element of the fam language code pattern rule-declaring element ‘(([] FunctionName Variable1 Variable2 .. VariableN)) [:=] G)’ , then this rule will not replace the element ‘TheElement’ .

Case 4 : If the element ‘([] FName Value1 Value2 .. ValueN)’ is in rule applying scope of (saved function-replacement rule1) and also in rule applying scope of (saved function-replacement rule 2) , .. , and also in rule applying scope of (saved function-replacement rule {k}) . And the element ‘([] FName Value1 Value2 .. ValueN)’ satisfy all replacement condition of all these rule .

In this case , among these saved function-replacement rule , the AI system will apply only 1 saved function-replacement rule , whose rule declaring event relative universal inherent index is highest logical executing order , to the element ‘([] FName Value1 Value2 .. ValueN)’ .

Case4 : in all other case , except Case 1 , Case 2 , Case 3 , Case 4 , then the AI system will replace the element ‘([] FName Value1 Value2 .. ValueN)’ by the element ‘G’ , and immediately the AI system update keep delay update the element G , and then the AI system consequently apply {N} in-line replacement (Variable1 -> updatedValue1 , Variable2 -> updatedValue2 , .. , VariableN -> updatedValueN) for all element inside the element ‘G’ , as a result the AI system update the element ‘G’ into the element ‘updatedG’ , and then the AI system can update the element ‘updatedG’ into the element ‘nextUpdatedG’ , and finally the AI system replace the original element ‘([] FName Value1 Value2 .. ValueN)’ by the element ‘nextUpdatedG’ .

For example :

Code :

(A (([] myFunction x y z) [:=] (x + y + z)) ([] myFunction 1 2 3) ABC)

Result must be : (A (([] myFunction x y z) [:=] (x + y + z)) (1 + 2 + 3) ABC)

For example :

Code : ((x [=] 3) (([] myFunction x y z ) [:=] (x 11 y 11 z)) x ([] myFunction A B C) x)

Result must be : ((x [=] 3) (([] myFunction x y z ) [:=] (x 11 y 11 z)) 3 (A 11 B 11 C) 3)

**3.1.5.3/ Fam language specific (function-replacement-in-scope (rule declaring)) rule :**

**Precaution :** It is not an element assignment operation , also not temporary element storing .. It is simply a rule to directly replace element by other specified element in fam language codepage .

The fam language specific code pattern :

**(([] FunctionName Variable1 Variable2 .. VariableN) [:=] G (M P))**

Where the element ‘[]’ , ‘[:=]’ are the fam language keyword element ; the element ‘FunctionName’ , ‘Variable1’ , ‘Variable2’ , .. , ‘VariableN’ are arbitrary element ; the element ‘G’ is arbitrary element ; element ‘M’ and ‘P’ are arbitrary element .

These specific code pattern element always begin with parenthese ‘(‘ and end with parenthese ‘)’ , always have strictly 4 nested element at 1-level of its own hierarchy structure , with space between them , :

‘([] FunctionName Variable1 Variable2 .. VariableN)’ , ‘[:=]’ , ‘G’ , ‘(M P)’

The first 1-level nested element ‘([] FunctionName Variable1 Variable2 .. VariableN)’ must begin with parenthese ‘(‘ and end with parenthese ‘)’ , must have the first 1-level nested element ‘[]’ , must have second 1-level nested element .

The nested element ‘(M P)’ always begin with parenthese ‘(‘ and end with parenthese ‘)’ , and have strictly 2 1-level nested element ‘M’ and ‘P’ with space between them .

Now given the element ‘FunctionName’ , ‘Variable1’ , ‘Variable2’ , .. , ‘VariableN’ , and element ‘M’ and element ‘N’ .

The fam language specific code pattern element : **(([] FunctionName Variable1 Variable2 .. VariableN) [:=] G (M P))**

Now describe the fam language specific strict instruction for AI system to interpret the fam language specific code pattern element ‘(([] FunctionName Variable1 Variable2 .. VariableN) [:=] G (M P))’.

This is fam language specific rule-declaring element , whose functionality is absolute analogous to the fam language specific function-replacement function-declaring element ‘(([] FunctionName Variable1 Variable2 .. VariableN) [:=] G)’ , with the only difference is that there is specified rule applying scope in this case .

When the AI system meet such fam language code pattern element , then the AI system do not update all element , except the 2 element ‘M’ and ‘P’ inside the element ‘(M P)’ , inside the element ‘(([] FunctionName Variable1 Variable2 .. VariableN) [:=] G (M P))’ , by applying other fam language replacement rule . But the AI system update the element ‘M’ and element ‘P’ by applying other fam language replacement rule , as a result : the element ‘M’ -> the element ‘updatedM’ , and the element ‘P’ -> the element ‘updatedP’ .

If the element ‘updatedM’ is not natural integer greater than 0 , or the element ‘updatedP’ is not natural integer greater than 0 , then the AI system replace the element ‘(([] FunctionName Variable1 Variable2 .. VariableN) [:=] G (M P))’ by the element ‘()’ .

If the element ‘updatedM’ is natural integer greater than 0 , and the element ‘updatedP’ is natural integer greater than 0 , and the AI system can not define the {updatedM}-level containing element of the element ‘(([] FunctionName Variable1 Variable2 .. VariableN) [:=] G (M P))’ , then the AI system take the

updated fam language codepage element as the temporary {updatedM}-level containing element of the element ‘(([] FunctionName Variable1 Variable2 .. VariableN) [:=] G (M P))’ only for this case .

If the element ‘updatedM’ is natural integer greater than 0 , and the element ‘updatedP’ is natural integer greater than 0 , and the AI system can define the {updatedM}-level containing element of the element ‘(([] FunctionName Variable1 Variable2 .. VariableN) [:=] G (M P))’ , then the AI replace the element ‘(([] FunctionName Variable1 Variable2 .. VariableN) [:=] G (M P))’ by the element ‘(([] FunctionName Variable1 Variable2 .. VariableN) [:=] G (updatedM updatedP))’ , and the AI system add a new rule (function-replacement) together with new rule applying scope , which is the set of all contained element of the {updatedM}-level containing element of the element ‘(([] FunctionName Variable1 Variable2 .. VariableN) [:=] G (M P))’ and in all level from 1 to {updatedM + updatedP} of the own hierarchy structure of the {updatedM}-level containing element of the element ‘(([] FunctionName Variable1 Variable2 .. VariableN) [:=] G (M P))’ .

Rule declaring event inherent index of this rule is the inherent index of the element ‘(([] FunctionName Variable1 Variable2 .. VariableN) [:=] G (M P))’ . Thus there are rule declaring event local inherent index of this rule , rule declaring event {n}-deep region inherent index of this rule , rule declaring event relative universal inherent index of this rule .

For example :

Code : (12 (([] myFunction A B) [:=] (A xx B)) ([] myFunction X Y) 11) ([] myFunction X Y)

Result must be : (12 (([] myFunction A B) [:=] (A xx B)) (X xx Y) 11) ([] myFunction X Y)

Code : (12 (([] myFunction A B) [:=] (A xx B) (2 3)) ([] myFunction X Y) 11) ([] myFunction X Y)

Result : (12 (([] myFunction A B) [:=] (A xx B) (2 3)) (X xx Y) 11) (X xx Y)

**3.1.5.4/ Fam language specific (function-instant-replacement (rule declaring)) rule :**

**Precaution :** It is not an element assignment operation , also not temporary element storing .. It is simply a rule to directly replace element by other specified element in fam language codepage .

The fam language code pattern function-declaring element :

**(([] FunctionName Variable1 Variable2 .. VariableN) [:=:] G)**

When the AI system meet such fam language function-declaring element , then the AI system update the element ‘G’ by other fam language replacement rule , as a result : the element ‘G’ -> the element ‘updatedG’ . And then the AI system add new function-replacement rule , which is corresponding to the fam language specific code pattern function-replacement function-declaring element ‘**(([] FunctionName Variable1 Variable2 .. VariableN) [:=] updatedG)’** .

**3.1.5.5/ Fam language specific (function-instant-replacement-in-scope (rule declaring)) rule :**

**Precaution :** It is not an element assignment operation , also not temporary element storing .. It is simply a rule to directly replace element by other specified element in fam language codepage .

The fam language code pattern function-declaring element :

**(([] FunctionName Variable1 Variable2 .. VariableN) [:=:] G (M P))**

When the AI system meet such fam language function-declaring element , then the AI system update the element ‘G’ by other fam language replacement rule , as a result : the element ‘G’ -> the element ‘updatedG’ . And then the AI system add new function-replacement rule , which is corresponding to the fam language specific code pattern function-replacement function-declaring element ‘**(([] FunctionName Variable1 Variable2 .. VariableN) [:=] updatedG (M P))’** .

## 3.1.6/ Fam language specific strict instruction for interpreting : fam language specific absolute-identical-equality-replacement rule :

The fam language specific code pattern : **(element1 [==] element2)**

Where element ‘element1’ and element ‘element2’ are arbitrary element , the element ‘[==]’ is fam language keyword element .

The specific code pattern element ‘(element1 [==] element2)’ always begin with parenthese ‘(‘ and end with parenthese ‘)’ , always have strictly 3 nested element at 1-level of its own hierarchy structure : ‘element1’ , ‘[==]‘ , ‘element2’ , with space between them .

Now given the element ‘element1’ and given the element ‘element2’ .

The fam language specific code pattern element : **(element1 [==] element2)**

Now describe the fam language specific strict instruction for AI system to interpret the fam language specific code pattern element ‘(element1 [==] element2)’ .

When the AI system meet the such fam language code pattern element ‘(element1 [==] element2)’ in the current fam language codepage , then the AI system firstly update the element ‘element1’ and the element ‘element2’ by applying other fam language specific rule (fam language specific replacement rule) to update : the element ‘element1’ -> the element ‘updatedElement1’ , and the element ‘element2’ -> the element ‘updatededElement2’ . Then the AI system replace the element ‘(element1 [==] element2)’ by the element ‘[1?]’ , which is the Boolean Logic True in fam language, if the element ‘updatedElement1’ and the element ‘updatedElement2’ are absolute identical equal , and by the element ‘[0?]’ , which is the Boolean Logic False in fam language , if the element ‘updatedElement1’ and the element ‘updatedElement2’ are not absolute identical equal . Thus the AI system updated the current fam language codepage into updated fam language codepage .

The element ‘[1?]’ , and element ‘[‘0?’]’ are fam language keyword element .

For example :

Fam language code page : (this element is not equal to any element) (element2) (someElement (this element is not equal to any element))

In this example , the first element ‘(this element is not equal to any element)’ is absolute identical equal to the second element ‘(this element is not equal to any element)’ .

For example :

Fam language message : ( (3 – 0) = someElement) ((3) otherElement)

In this example, in this loaded fam language code page, the element ‘(3 - 0)’ have its writing-element-description-code : “(3 - 0)”. The element ‘(3)’ have its writing-element-description-code : “(3)” . These writing-element-description-code are different , thus the fam language specific absolute identical equality interpreting rule tell that these element are not absolute identical equal. But in much many user context case, most AI system can reckon that these element ‘(3 - 0)’ and element ‘(3)’ are equal or identical equal. It is ok, and it do not conflict with fam language design principle.

For example :

Fam language message : ( (3 – 0) = someElement) ((3- 0) otherElement)

In this example, in this loaded fam language code page, the element ‘(3 - 0)’ have its writing-element-description-code : “(3 - 0)”. The element ‘(3- 0)’ have its writing-element-description-code : “(3- 0)” . Note about space inside these element. These writing-element-description-code are different, thus the fam language specific absolute identical equality interpreting rule tell these element are not absolute identical equal . But in much many user context case, most AI system can reckon that these element ‘(3 - 0)’ and element ‘(3- 0)’ are equal or identical equal. It is ok, and it do not conflict with fam language design principle.

### 3.1.7/ Fam language specific strict instruction for interpreting : fam language specific equivalent-belonging-replacement rule :

The fam language specific code pattern : **(elementA [<:] elementB)**

Where element ‘elementA’ and element ‘elementB’ are arbitrary element , element ‘[<:]’ is fam language keyword element .

These specific code pattern element always begin with parenthese ‘(‘ and end with parenthese ‘)’ , always have strictly 3 nested element at 1-level of its own hierarchy structure , with space between them .

Now given the element ‘elementA’ , give the element ‘elementB’ .

There is fam language specific code pattern element : **(elementA [<:] elementB)**

The AI system interpret this specific code pattern element ‘(elementA [<:] elementB)’ by the fam language specific (equivalent-belonging) rule :

When the AI system meet the such fam language specific code pattern element ‘(elementA [<:] elementB)’ in the current fam language codepage , then the AI system firstly update the element ‘elementA’ and the element ‘elementB’ by applying other fam language specific rule (fam language specific replacement rule) to update : the element ‘elementA’ -> the element ‘updatedElementA’ , and the element ‘elementB’ -> the element ‘updatedElementB’ . Then the AI system replace the element ‘(elementA [<:] elementB)’ by the element ‘[?1]’ (it is fam language Boolean Logic True) if the AI system see that in the 1-level of the own hierarchy structure of the element ‘updatedElementB’ there is at least 1 nested element , which is absolute identical equal to the element ‘updatedElementA’ . The AI system replace the element ‘(elementA [<:] elementB)’ by the element [0?]’ (it is the fam language Boolean Logic False) if the AI system see that in the 1-level of the own hierarchy structure of the element ‘updatedElementB’ there is no any nested element , which is absolute identical equal to the element ‘updatedElementA’ .

Thus the AI system just already updated the fam language codepage .

#### 3.1.7.1/ Fam language specific strict instruction for interpreting : fam language specific equivalent-belonging-№2-replacement rule :

The fam language specific code pattern : **(elementA [<::] elementB)**

Where element ‘elementA’ and element ‘elementB’ are arbitrary element , element ‘[<::]’ is fam language keyword element .

In these code pattern , the element ‘elementA’ and the element ‘elementB’ can be arbitrary element , the element ‘[<::]’ is fam language specific keyword element .

These specific code pattern element always begin with parenthese ‘(‘ and end with parenthese ‘)’ , always have strictly 3 nested element at 1-level of its own hierarchy structure , with space between them .

Now given the element ‘elementA’ , given the element ‘elementB’ .

There is fam language specific code pattern element : **(elementA [<::] elementB)**

The AI system interpret this specific code pattern element ‘(elementA [<::] elementB)’ by the fam language specific (equivalent-belonging-№2-replacement) rule :

When the AI system meet the such fam language code pattern element ‘(elementA [<::] elementB)’ in the current fam language codepage , then the AI system firstly update the element ‘elementA’ and the element ‘elementB’ by applying other fam language specific rule (fam language specific replacement rule) to update : the element ‘elementA’ -> the element ‘updatedElementA’ , and the element ‘elementB’ -> the element ‘updatedElementB’ . Then the AI system replace the element ‘(elementA [<::] elementB)’ by the element ‘[?1?]’ (it is the fam language Boolean Logic True) if the AI system see that in the own hierarchy structure of the element ‘updatedElementB’ there is at least 1 element , which is absolute identical equal to the element ‘updatedElementA’ . The AI system replace the element ‘(elementA [<::] elementB)’ by the element ‘[?0?]’ (it is the fam language Boolean Logic False) if the AI system see that in the own hierarchy structure of the element ‘updatedElementB’ there is no any element , which is absolute identical equal to the element ‘updatedElementA’ . Thus the AI system just already updated the fam language codepage .

## 3.1.8/ Fam language specific strict instruction for interpreting : fam language specific triple-Boolean-arithmetic rule :

Fam language specific instruction use triple Boolean arithmetic to control conditional execution and conditional interpretation .

#### 3.1.8.1/ Fam language specific strict instruction for interpreting : fam language specific triple-Boolean-value rule :

The fam language specific code pattern : **([?3?] TheElement)**

Where the element ‘TheElement’ is arbitrary element ; the element ‘[?3?]’ is fam language keyword element .

In this code pattern , the element ‘TheElement’ can be arbitrary element , the element ‘[?3?]’ is fam language specific keyword element .

These specific code pattern element always begin with parenthese ‘(‘ and end with parenthese ‘)’ , always have strictly 2 nested element at 1-level of its own hierarchy structure , with space between them .

Given the element ‘TheElement’ .

The fam language specific code pattern element : **‘([?3?] TheElement)’**

The AI system interpret this specific code pattern element ‘([?3?] TheElement)’ by the fam language specific (triple-Boolean-value) rule :

When the AI system meet the such fam language code pattern element ‘([?3?] TheElement)’ in the current fam language codepage , then the AI system firstly update the element ‘TheElement’ by applying other fam language specific rule (fam language specific replacement rule) to update : the element ‘TheElement’ -> the element ‘updatedTheElement’ . If the AI system reckon that the element ‘updatedTheElement’ is a standard Boolean Logic True element , then the AI system replace the element ‘([?3?] TheElement)’ by the element ‘[1?]’ , which is fam language specific Boolean Logic True . If the AI system reckon that the element ‘updatedTheElement’ is a standard Boolean Logic False , then the AI system replace the element ‘([?3?] TheElement)’ by the element ‘[0?]’ , which is fam language specific Boolean Logic False element .

Besides , if the AI system see that the element ‘updatedTheElement’ is absolute identical equal to the element ‘[0?]’ , then the AI system must definitely replace the element ‘([?3?] TheElement)’ by the element ‘[0?]’ . If the AI system see that the element ‘updatedTheElement’ is absolute identical equal to the element ‘[1?]’ , then the AI system must definitely replace the element ‘([?3?] TheElement)’ by the element ‘[1?]’ .

In any other case , which are not the above 2 case , then the AI system replace the element ‘([?3?] TheElement)’ by the element ‘()’ .

Thus the AI system update the fam language codepage .

For example :

**3.1.8.2/ Fam language specific strict instruction for interpreting : fam language specific triple-Boolean-value-negation-replacement rule :**

The fam language specific code pattern : **([!?] theElement)**

Where the element ‘theElement’ is arbitrary element ; the element ‘[!?]’ is fam language keyword element .

In this code pattern , the element ‘theElement’ can be arbitrary element , the element ‘[!?]’ is fam language specific keyword element .

These specific code pattern element always begin with parenthese ‘(‘ and end with parenthese ‘)’ , always have strictly 2 nested element at 1-level of its own hierarchy structure , with space between them .

Given the element ‘theElement’ .

The fam language specific code pattern element : **‘([!?] theElement)’**

The AI system interpret this specific code pattern element ‘([!?] theElement)’ by the fam language specific (triple-Boolean-value-negation-replacement) rule :

When the AI system meet the such fam language code pattern element ‘([!?] theElement)’ in the current fam language codepage , then the AI system firstly update the element ‘theElement’ by applying other fam language specific rule (fam language specific replacement rule) to update : the element ‘theElement’ -> the element ‘updatedTheElement’ . If the AI system reckon that the element ‘updatedTheElement’ is a standard Boolean Logic True element, then the AI system replace the element ‘([!?] theElement)’ by the element ‘[0?]’ , which is the fam language specific Boolean Logic False . If the AI system reckon that the element ‘updatedTheElement’ is a standard Boolean Logic False element, then the AI system replace the element ‘([!?] theElement)’ by the element ‘[1?]’ , which is the fam language specific Boolean Logic True . Besides , if the AI system see that the element ‘updatedTheElement’ is absolute identical equal to the element ‘[0?]’ , then the AI system must definitely replace the element ‘([!?] theElement)’ by the element ‘[1?]’ . If the AI system see that the element ‘updatedTheElement’ is absolute identical equal to the element ‘[1?]’ , then the AI system must definitely replace the element ‘([!?] theElement)’ by the element ‘[0?]’ .

In any other case , which are not the above 2 case , then the AI system replace the element ‘([!?] theElement)’ by the element ‘()’ . Thus the AI system update the fam language codepage .

**3.1.8.3/ / Fam language specific strict instruction for interpreting : fam language specific triple-Boolean-union rule :**

The fam language specific code pattern : **(AA [+?] BB)**

Where the element ‘AA’ and the element ‘BB’ are arbitrary element ; the element ‘[+?]’ is fam language keyword element .

These specific code pattern element always begin with parenthese ‘(‘ and end with parenthese ‘)’ , always have strictly 3 nested element at 1-level of its own hierarchy structure , with space between them .

Given the element ‘AA’ and element ‘BB’ .

The fam language specific code pattern element : **(AA [+?] BB)**

The AI system interpret this fam language specific code pattern element ‘(AA [+?] BB)’ by the fam language specific (triple-Boolean-union) rule :

When the AI system meet the such fam language code pattern element ‘(AA [+?] BB)’ in the current fam language codepage , then the AI system firstly update the element ‘AA’ and the element ‘BB’ by applying other fam language specific rule (fam language specific replacement rule) to update : the element ‘AA’ -> the element ‘updatedAA’ , the element ‘BB’ -> the element ‘updatedBB’ . If the AI system see that the element ‘updatedAA’ is not Boolean Logic Value ‘(0?)’ or ‘(1?)’ , or the element ‘updatedBB’ is not Boolean Logic Value ‘(0?)’ or ‘(1?)’ , then the AI system replace the element ‘(AA [+?] BB)’ by the element ‘()’ .

If the AI system see that the element ‘updatedAA’ is a Boolean Logic Value ‘(0?)’ or ‘(1?)’ , and the element ‘updatedBB’ is Boolean Logic Value ‘(0?)’ or ‘(1?)’ , then the AI system replace the element ‘(AA [+?] BB)’ by the result of standard Boolean Logic Union operation between 2 Boolean Logic Value ‘updatedAA’ and ‘updatedBB’ .

Thus the AI system update the fam language codepage .

**3.1.8.4/ Fam language specific strict instruction for interpreting : fam language specific triple-Boolean-distraction rule :**

The fam language specific code pattern : **(AA [-?] BB)**

Where the element ‘AA’ and the element ‘BB’ are arbitrary element ; the element ‘[-?]’ is fam language keyword element .

These specific code pattern element always begin with parenthese ‘(‘ and end with parenthese ‘)’ , always have strictly 3 nested element at 1-level of its own hierarchy structure , with space between them .

Given the element ‘AA’ and element ‘BB’ .

The fam language specific code pattern element : **(AA [-?] BB)**

The AI system interpret this fam language specific code pattern element ‘(AA [-?] BB)’ by the fam language specific (triple-Boolean-distraction) rule :

When the AI system meet the such fam language code pattern element ‘(AA [-?] BB)’ in the current fam language codepage , then the AI system firstly update the element ‘AA’ and the element ‘BB’ by applying other fam language specific rule (fam language specific replacement rule) to update : the element ‘AA’ -> the element ‘updatedAA’ , the element ‘BB’ -> the element ‘updatedBB’ . If the AI system see that the element ‘updatedAA’ is not Boolean Logic Value ‘(0?)’ or ‘(1?)’ , or the element ‘updatedBB’ is not Boolean Logic Value ‘(0?)’ or ‘(1?)’ , then the AI system replace the element ‘(AA [-?] BB)’ by the element ‘()’ .

If the AI system see that the element ‘updatedAA’ is a Boolean Logic Value ‘(0?)’ or ‘(1?)’ , and the element ‘updatedBB’ is Boolean Logic Value element ‘(0?)’ or ‘(1?)’, then the AI system replace the element ‘(AA [-?] BB)’ by the result of standard Boolean Logic Distraction operation between 2 Boolean Logic Value ‘updatedAA’ and ‘updatedBB’ .

Thus the AI system update the fam language codepage .

**3.1.8.5/ Fam language specific strict instruction for interpreting : fam language specific triple-Boolean-intersection rule :**

The fam language specific code pattern : **(AA [&?] BB)**

Where the element ‘AA’ and the element ‘BB’ are arbitrary element ; the element ‘[&?]’ is fam language keyword element .

These specific code pattern element always begin with parenthese ‘(‘ and end with parenthese ‘)’ , always have strictly 3 nested element at 1-level of its own hierarchy structure , with space between them .

Given the element ‘AA’ and element ‘BB’ .

The fam language specific code pattern element : **(AA [&?] BB)**

The AI system interpret this fam language specific code pattern element ‘(AA [&?] BB)’ by the fam language specific (triple-Boolean-intersection) rule :

When the AI system meet the such fam language code pattern element ‘(AA [&?] BB)’ in the current fam language codepage , then the AI system firstly update the element ‘AA’ and the element ‘BB’ by applying other fam language specific rule (fam language specific replacement rule) to update : the element ‘AA’ -> the element ‘updatedAA’ , the element ‘BB’ -> the element ‘updatedBB’ . If the AI system see that the element ‘updatedAA’ is not Boolean Logic Value ‘(0?)’ or ‘(1?)’ , or the element ‘updatedBB’ is not Boolean Logic Value ‘(0?)’ or ‘(1?)’ , then the AI system replace the element ‘(AA [&?] BB)’ by the element ‘()’ .

If the AI system see that the element ‘updatedAA’ is a Boolean Logic Value ‘(0?)’ or ‘(1?)’, and the element ‘updatedBB’ is Boolean Logic Value ‘(0?)’ or ‘(1?)’ , then the AI system replace the element ‘(AA [&?] BB)’ by the result of standard Boolean-Logic-Intersection operation between 2 Boolean Logic Value ‘updatedAA’ and ‘updatedBB’ .

Thus the AI system update the fam language codepage .

**3.1.8.6/ Fam language specific strict instruction for interpreting : fam language specific triple-Boolean-implication rule :**

The fam language specific code pattern : **(AA [->?] BB)**

Where the element ‘AA’ and the element ‘BB’ are arbitrary element ; the element ‘[->?]’ is fam language keyword element .

These specific code pattern element always begin with parenthese ‘(‘ and end with parenthese ‘)’ , always have strictly 3 nested element at 1-level of its own hierarchy structure , with space between them .

Given the element ‘AA’ and element ‘BB’ .

The fam language specific code pattern element : **(AA [->?] BB)**

The AI system interpret this fam language specific code pattern element ‘(AA [->?] BB)’ by the fam language specific (triple-Boolean-implication) rule :

When the AI system meet the such fam language code pattern element ‘(AA [->?] BB)’ in the current fam language codepage , then the AI system firstly update the element ‘AA’ and the element ‘BB’ by applying other fam language specific rule (fam language specific replacement rule) to update : the element ‘AA’ -> the element ‘updatedAA’ , the element ‘BB’ -> the element ‘updatedBB’ . If the AI system see that the element ‘updatedAA’ is not Boolean Logic Value ‘(0?)’ or ‘(1?)’ , or the element ‘updatedBB’ is not Boolean Logic Value ‘(0?)’ or ‘(1?)’ , then the AI system replace the element ‘(AA [->?] BB)’ by the element ‘()’ .

If the AI system see that the element ‘updatedAA’ is a Boolean Logic Value ‘(0?)’ or ‘(1?)’ , and the element ‘updatedBB’ is Boolean Logic Value ‘(0?)’ or ‘(1?)’ , then the AI system replace the element ‘(AA [->?] BB)’ by the result of standard Boolean-Logic-Implication operation between 2 Boolean Logic Value ‘updatedAA’ and ‘updatedBB’ .

Thus the AI system update the fam language codepage .

## 3.1.9 / Fam language specific strict instruction for interpreting : fam language specific ordered-set-arithmetic rule :

Fam language ordered-set-arithmetic is to manipulate arithmetic operation of order set of finite number of element .

Each fam language can simulate a abstract ordered set , in which all 1-level nested element of the original fam language element can simulate element of the abstract ordered set . An abstract ordered set can have no any element .

**3.1.9.1 / Fam language specific strict instruction for interpreting : fam language specific ordered-set-union rule :**

The fam language specific code pattern : **(AA [+] BB)**

Where the element ‘AA’ and the element ‘BB’ are arbitrary element ; the element ‘[+]’ is fam language keyword element .

These specific code pattern element always begin with parenthese ‘(‘ and end with parenthese ‘)’ , always have strictly 3 nested element at 1-level of its own hierarchy structure , with space between them .

Given the element ‘AA’ and element ‘BB’ .

The fam language specific code pattern element : **(AA [+] BB)**

The AI system interpret this fam language specific code pattern element ‘(AA [+] BB)’ by the fam language specific (ordered-set-union) rule :

When the AI system meet the such fam language code pattern element ‘(AA [+] BB)’ in the current fam language codepage , then the AI system firstly update the element ‘AA’ and the element ‘BB’ by applying other fam language specific rule (fam language specific replacement rule) to update : the element ‘AA’ -> the element ‘updatedAA’ , the element ‘BB’ -> the element ‘updatedBB’ . If the element ‘updatedAA’ have no any 1-level nested element , and the element ‘updatedBB’ have no any 1-level nested element , then the AI system replace the element ‘(AA [+] BB)’ by the element ‘()’ . If the element ‘updatedAA’ have 1-level nested element , or the element ‘updatedBB’ have 1-level nested element , then the AI system replace the element ‘(AA [+] BB)’ by the result element , which contain only 1-level nested element of the element ‘updatedAA’ and 1-level nested element of the element ‘updatedBB’ , in the place order that all1-level nested element of element ‘updatedAA’ stay before all 1-level nested element of the element ‘updatedBB’ . Thus the AI system update the fam language codepage .

The fam language code pattern element : **(A1 [+] A2 [+] A3 .. [+] An)** is analogous to the fam language code pattern element (A1 [+] A2) .

**3.1.9.2 / Fam language specific strict instruction for interpreting : fam language specific ordered-set-distraction rule :**

The fam language specific code pattern : **(AA [-] BB)**

Where the element ‘AA’ and the element ‘BB’ are arbitrary element ; the element ‘[-]’ is fam language keyword element .

These specific code pattern element always begin with parenthese ‘(‘ and end with parenthese ‘)’ , always have strictly 3 nested element at 1-level of its own hierarchy structure , with space between them .

Given the element ‘AA’ and element ‘BB’ .

The fam language specific code pattern element : **(AA [-] BB)**

The AI system interpret this fam language specific code pattern element ‘(AA [-] BB)’ by the fam language specific (ordered-set-distraction) rule :

When the AI system meet the such fam language code pattern element ‘(AA [-] BB)’ in the current fam language codepage , then the AI system firstly update the element ‘AA’ and the element ‘BB’ by applying other fam language specific rule (fam language specific replacement rule) to update : the element ‘AA’ -> the element ‘updatedAA’ , the element ‘BB’ -> the element ‘updatedBB’ . If the element ‘updatedAA’ have no any 1-level nested element , then the AI system replace the element ‘(AA [-] BB)’ by the element ‘()’ . If the element ‘updatedAA’ have all its 1-level nested element , which are equivalent belonging to the element ‘updatedBB’ , then the AI system replace the element ‘(AA [-] BB)’ by the element ‘()’ . If the element ‘updatedAA’ contain , at least , 1-level element , which is not equivalent belonging to the element ‘updatedBB’ , then the AI system replace the element ‘(AA [-] BB)’ by the result element , which contain all 1-level nested element of the element ‘updatedAA’ with the condition that such 1-level nested element are not equivalent belonging to the element ‘updatedBB’ , in the place order that all1-level nested element of element ‘updatedAA’ stay before all 1-level nested element of the element ‘updatedBB’ . Thus the AI system update the fam language codepage .

**3.1.9.3 / Fam language specific strict instruction for interpreting : fam language specific ordered-set-intersection rule :**

The fam language specific code pattern : **(AA [&] BB)**

Where the element ‘AA’ and the element ‘BB’ are arbitrary element ; the element ‘[&]’ is fam language keyword element .

These specific code pattern element always begin with parenthese ‘(‘ and end with parenthese ‘)’ , always have strictly 3 nested element at 1-level of its own hierarchy structure , with space between them .

Given the element ‘AA’ and element ‘BB’ .

The fam language specific code pattern element : **(AA [&] BB)**

The AI system interpret this fam language specific code pattern element ‘(AA [-] BB)’ by the fam language specific (ordered-set-intersection) rule :

When the AI system meet the such fam language code pattern element ‘(AA [&] BB)’ in the current fam language codepage , then the AI system firstly update the element ‘AA’ and the element ‘BB’ by applying other fam language specific rule (fam language specific replacement rule) to update : the element ‘AA’ -> the element ‘updatedAA’ , the element ‘BB’ -> the element ‘updatedBB’ . If the element ‘updateAA’ have no any 1-level nested element , or the element ‘updatedBB’ have no any 1-level nested element , then the AI system replace the element ‘(AA [&] BB)’ by the element ‘()’ . If the element ‘updatedAA’ have no any 1-level nested element , which are equivalent belonging to the element ‘updatedBB’ , then the AI system replace the element ‘(AA [-] BB)’ by the element ‘()’ . If the element ‘updatedAA’ have 1-level nested element , which are equivalent belonging to the element ‘updatedBB’ , then the AI system replace the element ‘(AA [-] BB)’ by the result element , which contain all 1-level nested element of the element ‘updatedAA’ with the condition that such 1-level nested element are equivalent belonging to the element ‘updatedBB’ , in the place order that all1-level nested element of element ‘updatedAA’ stay before all 1-level nested element of the element ‘updatedBB’ .

Thus the AI system update the fam language codepage .

**3.1.9.4 / Fam language specific strict instruction for interpreting : fam language specific ordered-set-multiplication-on-natural-number rule :**

The fam language specific code pattern : **(AA [*] N)**

Where the element ‘AA’ and the element ‘N’ are arbitrary element ; the element ‘[*]’ is fam language keyword element .

These specific code pattern element always begin with parenthese ‘(‘ and end with parenthese ‘)’ , always have strictly 3 nested element at 1-level of its own hierarchy structure , with space between them .

Given the element ‘AA’ and element ‘N’ .

The fam language specific code pattern element : **(AA [*] N)**

The AI system interpret this fam language specific code pattern element ‘(AA [*] N)’ by the fam language specific (ordered-set-multiplication-on-natural-number) rule :

When the AI system meet the such fam language code pattern element ‘(AA [*] N)’ in the current fam language codepage , then the AI system firstly update the element ‘AA’ and the element ‘N’ by applying other fam language specific rule (fam language specific replacement rule) to update : the element ‘AA’ -> the element ‘updatedAA’ , the element ‘N’ -> the element ‘updatedN’ . If the element ‘updatedAA’ have no any 1-level nested element , then the AI system replace the element ‘(AA [*] N)’ by the element ‘()’ . If the element ‘updatedN’ is not natural number , which is greater than 0 , then the AI system replace the element ‘(AA [*] N)’ by the element ‘()’ . If the element ‘updatedN’ is natural number greater than 0 and the element ‘updatedAA’ have 1-level nested element , then AI system replace the element ‘(AA [*] N)’ by the result element , which contain all 1-level nested element of the element ‘updatedAA’ and in repeating n time . Thus the AI system update the fam language codepage .

**3.1.9.5 / Fam language specific strict instruction for interpreting : fam language specific (sequence-of-natural-number) rule :**

The fam language specific code pattern : **(N1 [..] N2)**

Where the element ‘N1’ and the element ‘N2’ are arbitrary element ; the element ‘[..]’ is fam language keyword element .

These specific code pattern element always begin with parenthese ‘(‘ and end with parenthese ‘)’ , always have strictly 3 nested element at 1-level of its own hierarchy structure , with space between them .

Given the element ‘N1’ and element ‘N2’ .

The fam language specific code pattern element : **(N1 [..] N2)**

The AI system interpret this fam language specific code pattern element ‘(N1 [..] N2)’ by the fam language specific (sequence-of-natural-number) rule :

When the AI system meet the such fam language code pattern element ‘(N1 [..] N2)’ in the current fam language codepage , then the AI system firstly update the element ‘N1’ and the element ‘N2’ by applying other fam language specific rule (fam language specific replacement rule) to update : the element ‘N1’ -> the element ‘updatedN1’ , the element ‘N2’ -> the element ‘updatedN2’ . If the element ‘updatedN1’ is not natural number , or the element ‘updatedN2’ is not natural number , then the AI system replace the element ‘(N1 [..] N2)’ by the element ‘()’ . If the element ‘updatedN1’ is natural number , and the element

‘updatedN2’ is natural number , and the element ‘updatedN1’ as natural number > the element ‘updatedN2’ as natural number , then the AI system replace the element ‘(N1 [..] N2)’ by the element ‘()’ . If the element ‘updatedN1’ is natural number , and the element ‘updatedN2’ is natural number , and the element ‘updatedN1’ as natural number is smaller or equal to the element ‘updatedN2’ as natural number , then the AI system replace the element ‘(N1 [..] N2)’ by the element ‘(I1 I2 .. Ik) , where I1 , I2 , .. , Ik are sequence of consecutive natural number range of from ‘updatedN1’ to ‘updatedN2’ .

Thus the AI system update the fam language codepage .

## 3.1.10 / Fam language specific strict instruction for interpreting : fam language specific (explicit unconditional set) rule :

The fam language specific code pattern : **([[]] FF [:] XX [<:] AA)**

Where the element ‘FF’ and the element ‘XX’ and the element ‘AA’ are arbitrary element ; the element ‘[:]’ and the element ‘[<:]’ and the element ‘[[]]’ are fam language keyword element . The element ‘[[]]’ is keyword to hint about explicit ordered set .

These specific code pattern element always begin with parenthese ‘(‘ and end with parenthese ‘)’ , always have strictly 6 nested element at 1-level of its own hierarchy structure , with space between them .

Given the element ‘FF’ and element ‘XX’ and element ‘AA’ .

The fam language specific code pattern element : **([[]] FF [:] XX [<:] AA)**

The AI system interpret this fam language specific code pattern element ‘(FF [:] XX [<:] AA)’ by the fam language specific (explicit unconditional set) rule :

When the AI system meet the such fam language code pattern element ‘([[]] FF [:] XX [<:] AA)’ in the current fam language codepage , then the AI system firstly update the element ‘FF’ and the element ‘XX’ and the element ‘AA’ by applying other fam language specific rule (fam language specific replacement rule) to update : the element ‘FF’ -> the element ‘updatedFF’ , the element ‘XX’ -> the element ‘updatedXX’ , the element ‘AA’ -> the element ‘updatedAA’ . If the element ‘updatedAA’ have no 1-level nested element , then the AI system replace the element ‘([[]] FF [:] XX [<:] AA)’ by the element ‘()’ .

If the element ‘updatedAA’ have 1-level nested element , for example these 1-level nested element ‘AAA1’ , ‘AAA2’ , .. , ‘AAAn’ , then the AI system replace the element ‘([[]] FF [:] XX [<:] AA)’ by the element ‘((updatedFF [|] updatedXX [->] AAA1) (updatedFF [|] updatedXX [->] AAA2) .. (updatedFF [|] updatedXX [->] AAAn))’ .

Thus the AI system update the fam language codepage .

## 3.1.11 / Fam language specific strict instruction for interpreting : fam language specific (explicit conditional set) rule :

The fam language specific code pattern : **([[]] X [:] X [<:] A [&] Condition)**

Where the element ‘X’ and the element ‘A’ and the element ‘Condition’ are arbitrary element ; the element ‘[:]’ and the element [<:] and the element [&] are fam language keyword element .

These specific code pattern element always begin with parenthese ‘(‘ and end with parenthese ‘)’ , always have strictly 8 nested element at 1-level of its own hierarchy structure , with space between them .

Given the element ‘X’ and element ‘A’ and element ‘Condition’ .

The fam language specific code pattern element : **([[]] X [:] X [<:] A [&] Condition)**

The AI system interpret this fam language specific code pattern element ‘([[]] X [:] X [<:] A [&] Condition)’ by the fam language specific (conditional set) rule :

When the AI system meet the such fam language code pattern element ‘([[]] X [:] X [<:] A [&] Condition)’ in the current fam language codepage , then the AI system firstly update the element ‘X’ and the element ‘A’ and the element ‘Condition’ by applying other fam language specific rule (fam language specific replacement rule) to update : the element ‘X’ -> the element ‘updatedX’ , the element ‘A’ -> the element ‘updatedA’ , the element ‘Condition’ -> the element ‘updatedCondition’ . If the element ‘updatedA’ have no 1-level nested element , then the AI system replace the element ‘([[]] X [:] X [<:] A [&] Condition)’ by the element ‘()’ .

If the element ‘updatedA’ have 1-level nested element , for example these 1-level nested element ‘AA{1}’ , ‘AA{2}’ , .. , ‘AA{n}’ , and all element ‘([?3?] (updatedCondition [|] updatedX [->] AA{1}))’ is not Boolean Logic True , and element ‘([?3?] (updatedCondition [|] updatedX [->] AA{2}))’ is not Boolean Logic True , and .. , and element ‘([?3?] (updatedCondition [|] updatedX [->] AA{n}))’ is not Boolean Logic True , then the AI system replace the element ‘([[]] X [:] X [<:] A [&] Condition)’ by the element ‘()’ . If there is at least 1 element among element ‘AA{1}’ , ‘AA{2}’ , .. , ‘AA{n}’ , for example element ‘AA{p}’ satisfy condition ‘([?3?] (updatedCondition [|] updatedX [->] AA{p}))’ is Boolean Logic True , then the AI system replace the element ‘([[]] X [:] X [<:] A [&] Condition)’ by the element ‘(AA{i1} AA{i2} .. AA{ik})’ , where {i1} < {i2} < {i3} .. < {ik} , and all element ‘AA{i1}’ , ‘AA{i2}’ , .. , ‘AA{ik}’ are 1-level nested element of the element ‘updatedA’ , and ([?3?] (updatedCondition [|] updatedX [->] AA{i1})) = Boolean logic True , ([?3?] (updatedCondition [|] updatedX [->] AA{i2})) = Boolean Logic True , .. , ([?3?] (updatedCondition [|] updatedX [->] AA{ik})) = Boolean Logic True .

In other explanation , the AI system replace the element ‘([[]] X [:] X [<:] A [&] Condition)’ by the result element , which contain all 1-level nested element ‘AA{iq}’ of the element ‘updatedA’ , in same place order of 1-level nested element of the element ‘updatedA’ , and with the condition that all such element ([?3?] (updatedCondition [|] updatedX [->] AA{iq})) are Boolean Logic True .

Thus the AI system update the fam language codepage .

## 3.1.12 / Fam language specific strict instruction for interpreting : fam language specific (conditional execution) rule :

The fam language specific code pattern : **([??] Condition [:] F)**

Where the element ‘Condition’ and the element ‘F’ are arbitrary element ; the element ‘[??]’ and the element [:] are fam language keyword element . The keyword element ‘[??]’ hint the ‘If clause’ .

These specific code pattern element always begin with parenthese ‘(‘ and end with parenthese ‘)’ , always have strictly 4 nested element at 1-level of its own hierarchy structure , with space between them .

Given the element ‘Condition’ and element ‘F’.

The fam language specific code pattern element : **([??] Condition [:] F)**

The AI system interpret this fam language specific code pattern element ‘([??] Condition [:] F)’ by the fam language specific (conditional-execution) rule :

When the AI system meet the such fam language code pattern element ‘([??] Condition [:] F)’ in the current fam language codepage , then the AI system firstly update the element ‘Condition’ by applying other fam language specific rule (fam language specific replacement rule) to update : the element ‘Condition’ -> the element ‘updatedCondition’ . If the element ‘([?3?] updatedCondition)’ is not Boolean Logic True , then the AI system replace the element ‘([??] Condition [:] F)’ by the element ‘()’ . If the element ‘([?3?] updatedCondition)’ is Boolean Logic True , then the AI system replace the element ‘([??] Condition [:] F)’ by the element ‘F’ .

Thus the AI system update the fam language codepage .

## 3.1.13 / Fam language specific strict instruction for interpreting : fam language specific (execution-looping) rule :

The fam language specific code pattern : **([***] X [<:] A [:] F)**

Where the element ‘X’ and the element ‘A’ and the element ‘F’ are arbitrary element ; the element ‘[<:]’ and the element ‘[:]’ and the element ‘[***]’ are fam language keyword element . The keyword element ‘[***]’ hint looping .

These specific code pattern element always begin with parenthese ‘(‘ and end with parenthese ‘)’ , always have strictly 6 nested element at 1-level of its own hierarchy structure , with space between them .

Given the element ‘X’ and element ‘A’ and element ‘F’

The fam language specific code pattern element : **([***] X [<:] A [:] F)**

The AI system interpret this fam language specific code pattern element ‘([***] X [<:] A [:] F)’ by the fam language specific (execution-looping) rule :

When the AI system meet the such fam language code pattern element ‘([***] X [<:] A [:] F)’ in the current fam language codepage , then the AI system firstly update the element ‘A’ by applying other fam language specific rule (fam language specific replacement rule) to update : the element ‘A’ -> the element ‘updatedA’ . If the element ‘updatedA’ have no any 1-level nested element , then the AI system replace the element ‘([***] X [<:] A [:] F)’ by the element ‘()’ .

If the element ‘updatedA’ have at least 1-level nested element , then the AI system firstly update the element ‘X’ and the element ‘F’ by applying other fam language specific rule (fam language specific replacement rule) to update : the element ‘X’ -> the element ‘updatedX’ , the element ‘F’ -> the element ‘updatedF’ , then the AI system do the following procedure of consecutive replacement :

+ The AI system take the first 1-level nested element of the element ‘updateA’ , for example the 1-level nested element AA1 . Then the AI system replace the element ‘([***] X [<:] A [:] F)’ by the 1 looping element ‘(updatedF [|] updatedX [->] AA1)’ , which have only 5 1-level nested element ‘updatedF’ , ‘[|]’ , ‘updatedX’ , ‘[->]’ ‘AA1’ . The AI system apply all fam language specific rule to execute this looping element . After finishing executing this looping element , then

+ The AI system replace the 5-th 1-level nested element of this looping element by the element ‘AA2’ , where the element ‘AA2’ is the second 1-level nested element of the element ‘updatedA’ . Thus this looping element become ‘(updatedF [|] updatedX [->] AA2)’ . The AI system apply all fam language specific rule to execute this looping element . After finishing executing this looping element , then

+ The AI system continue analogous step ..

+ The AI system reach to final step : The AI system replace the 5-th 1-level nested element of this looping element by the element ‘AA{n}’ , where the element ‘AA{n}’ is the last 1-level nested element of the element ‘updatedA’ . Thus this looping element become ‘(updatedF [|] updatedX [->] AA{n})’ . The AI system apply all fam language specific rule to execute this looping element . After finishing executing this looping element , then the AI system replace this looping element by the element ‘()’ .

Thus the AI system update the fam language codepage .

## 3.1.14/ Fam language specific strict instruction for interpreting rule : fam language specific accessing-by-single-index interpreting rule :

The fam language specific code pattern : **(typicalElement [{] indexOfNestedElement [}])**

Where element ‘typicalElement’ is arbitrary element ; element ‘indexOfNestedElement’ is arbitrary element; element ‘[{]’ and ‘[}]’ are fam language keyword element .

The specific code pattern element ‘(typicalElement [{] indexOfNestedElement [}])’ always begin with parenthese ‘(‘ and end with parenthese ‘)’ , always have 4 nested element at 1-level of its own hierarchy structure : ‘typicalElement’ , ‘[{]‘ , ‘indexOfNestedElement’ , ‘[}]’ with space between them .

Given element ‘typicalElement’ . Given element ‘indexOfNestedElement’ :

There is fam language specific code pattern element : **(typicalElement [{] indexOfNestedElement [}])**

The AI system interpret this specific code pattern element ‘(typicalElement [{] indexOfNestedElement [}])’ by the fam language specific (accessing-by-index) rule :

When the AI system meet the such fam language specific code pattern element

‘(typicalElement [{] indexOfNestedElement [}])’ in the current fam language codepage , then the AI system firstly update the element ‘typicalElement’ and the element ‘indexOfNestedElement’ by applying other fam language specific rule (fam language specific replacement rule) to update : the element ‘typicalElement’ -> the element ‘updatedTypicalElement’ , and the element ‘indexOfNestedElement’ -> the element ‘updatedIndexOfNestedElement’ .

If the AI system see that the element ‘updatedTypicalElement’ have no nested element , then the AI system replace the element ‘(typicalElement [{] indexOfNestedElement [}])’ by the element ‘()’ in the updated fam language codepage .

If the AI system reckon (by AI system knowledge in the using context) that the element ‘updatedIndexOfNestedElement’ is not integer in range of 0 to infinity , then the AI system replace the element ‘(typicalElement [{] indexOfNestedElement [}])’ by the element ‘()’ in the updated fam language codepage .

If the AI system reckon (by AI system knowledge in the using context) that the element ‘updatedIndexOfNestedElement’ is an integer , which is outside the range of all index of all 1-level nested element of the element ‘updatedTypicalElement’ , then the AI system replace the element ‘(typicalElement [{] indexOfNestedElement [}])’ by the element ‘()’ in the updated fam language codepage .

If the AI system reckon (by AI system knowledge in the using context) that the element ‘updatedIndexOfNestedElement’ is an integer , which is inside the range of all index of all 1-level nested

element of the element ‘updatedTypicalElement’ , then the AI system replace the element ‘(typicalElement [{] indexOfNestedElement [}])’ by the 1-level nested element of index ‘updatedIndexOfNestedElement’ of the element ‘updatedTypicalElement’ in the updated fam language codepage .

For example :

Code : ((element1 (element2 (element3 element4))) [{] 1 [}])

In this example, when AI system interpret this specific code pattern element ‘((element1 (element2 (element3 element4))) [{] 1 [}])’ , the AI system will take the element ‘(element2 (element3 element4))’ as temporary interpretation of the element ‘((element1 (element2 (element3 element4))) [{] 1 [}])’ .

Code : (((element1 (element2 (element3 element4))) [{] 1 [}]) [{] 0 [}])

In this example, when AI system interpret this specific code pattern element, the AI system take the element ‘(element2 (element3 element4))’ as the temporary interpretation of the element ‘((element1 (element2 (element3 element4))) [{] 1 [}])’ , and continue to interpret . Thus the AI system take the element ‘((element2 (element3 element4)) [{] 0 [}])’ as the 1-step temporary interpretation of the element ‘(((element1 (element2 (element3 element4))) [{] 1 [}]) [{] 0 [}])’ , and continue to interpret . Next the AI system interpret can take the element ‘element2’ as the temporary interpretation of the element ‘((element2 (element3 element4))) [{] 0 [}])’ . Thus the AI system take the element ‘element2’ as the 2-step temporary interpretation of the element ‘(((element1 (element2 (element3 element4))) [{] 1 [}]) [{] 0 [}])’ .

### 3.1.15/ Fam language specific strict instruction for interpreting rule : fam language specific accessing-by-multi-index interpreting rule .

The fam language specific code pattern : **(typicalElement [{] i1 i2 [}])**

The fam language specific code pattern : **(typicalElement [{] i1 i2 i3 [}])**

The fam language specific code pattern : **(typicalElement [{] i1 i2 i3 i4 [}])**

The fam language specific code pattern : **(typicalElement [{] i1 i2 i3 i4 i5 [}])**

And so on ..

These above fam language code pattern are used to specify fam language specific (assessing-by-multi-index) rule .

These rule are used to replace the fam language specific code pattern element by its specified nested element in the specified level of its own hierarchy structure .

For example :

The fam language code pattern : **(typicalElement [{] i1 i2 [}])**

Where element ‘typicalElement’ is arbitrary element ; element ‘i1’ , element ‘i2’ are arbitrary element; element ‘[{]’ and ‘[}]’ are fam language keyword element .

The specific code pattern element ‘(typicalElement [{] i1 i2 [}])’ always begin with parenthese ‘(‘ and end with parenthese ‘) , always have strictly 5 nested element at 1-level of its own hierarchy structure : ‘typicalElement’ , ‘[{]‘ , ‘i1’ , ‘i2’ , ‘[}]’ with space between them .

Given an element ‘typicalElement’ , given element ‘i1’ , given element ‘i2’ .

The fam language code pattern element : (typicalElement [{] i1 i2 [}]) .

The following illustrating equation can explain how AI system must use fam language specific instruction to interpret the element ‘(typicalElement [{] i1 i2 [}])’ :

The illustrating equation : (typicalElement [{] i1 i2 [}]) = ((typicalElement [{] i1 [}]) [{] i2 [}]) .

Thus in this case , the AI system firstly replace the element ‘(typicalElement [{] i1 i2 [}])’ by the element

‘((typicalElement [{] i1 [}]) [{] i2 [}])’ . The element ‘((typicalElement [{] i1 [}]) [{] i2 [}])’ is qualified fam language code pattern element of the fam language specific (accessing-by-single-index) rule . Thus the AI system next must apply fam language specific (accessing-by-single-index) rule to deal with the element ‘((typicalElement [{] i1 [}]) [{] i2 [}])’ .

In the analogous way , there are such illustrating equation to explain how AI system must use fam language specific instruction to interpret element of the above fam language specific pattern :

(typicalElement [{] i1 i2 i3 [}]) = ((typicalElement [{] i1 i2 [}]) [{] i3 [}])

(typicalElement [{] i1 i2 i3 i4 [}]) = ((typicalElement [{] i1 i2 i3 [}]) [{] i4 [}])

(typicalElement [{] i1 i2 i3 i4 i5 [}]) = ((typicalElement [{] i1 i2 i3 i4 [}]) [{] i5 [}])

(typicalElement [{] i1 i2 i3 i4 i5 i6 [}]) = ((typicalElement [{] i1 i2 i3 i4 i5 [}]) [{] i6 [}])

..

And so on ..

## 3.1.16/ Fam language specific strict instruction for interpreting : fam language specific default-accessing-by-single-index interpreting rule .

This rule is not the familiar containing-element-reference rule as general understanding .

This rule is the absolute fam language specific strict instruction for AI system to strictly follow to interpret element into temporary interpretation of element. It is the only meaning of this rule , and fam language , as an abstract language , do not specify any specific instruction for containing-element-reference in general meaning .

But by to apply this rule , the impact is so much analogous to general understanding familiar containing-element-reference .

The fam language specific code pattern : **([{] elementIndex [}])**

Where element ‘elementIndex’ is arbitrary element ; element ‘[{]’ and ‘[}]’ are fam language keyword element .

The fam language specific code pattern element ‘([{] elementIndex [}])’ always begin with parenthese ‘(‘ and end with parenthese ‘)’ , always have strictly 3 nested element at 1-level of its own hierarchy structure : ‘[{]‘ , ‘elementIndex’ , ‘[}]’ with space between them .

Given element ‘elementIndex’ :

There is fam language specific code pattern element : **([{] elementIndex [}])**

Lets temporary use the term ‘the primary containing element’ of the element ‘([{] elementIndex [}])’ to specify the element , which contain this specific code pattern element ‘([{] elementIndex [}])’ at the 1-level in the own hierarchy structure of this primary containing element .

When the AI system meet the such fam language specific code pattern element ‘([{] elementIndex [}])’ in the current fam language codepage , then the AI system firstly update the element ‘elementIndex’ by applying other fam language specific rule (fam language specific replacement rule) to update : the element ‘elementIndex’ -> the element ‘updatedElementIndex’ . Then

If the AI system reckon (by AI system knowledge in the using context) that the element ‘updatedElementIndex’ is not integer in range of 0 to infinity , then the AI system replace the element ‘([{] elementIndex [}])’ by the element ‘()’ in this primary containing element of the original element ‘([{] elementIndex [}])’ . Thus the AI system update the fam language codepage .

If the AI system reckon (by AI system knowledge in the using context) that the element ‘updatedElementIndex’ is an integer , which is outside the range of all index of all nested element at 1-level of the own hierarchy structure of the primary containing element of the element ‘([{] elementIndex [}])’ , then the AI system replace the element ‘([{] elementIndex [}])’ by the element ‘()’ in this primary containing element of the original element ‘([{] elementIndex [}])’ . Thus the AI system update the fam language codepage .

If the AI system reckon (by AI system knowledge in the using context) that the element ‘updatedElementIndex’ is an integer , which is inside the range of all index of all nested element at 1-level of the own hierarchy structure of the primary containing element of the original element ‘([{] elementIndex [}])’ , then the AI system replace the element ‘([{] elementIndex [}])’ by the nested element of index ‘updatedElementIndex’ of 1-level of the own hierarchy structure of (the primary containing element of the original element ‘([{] elementIndex [}])’) in the updated fam language codepage .

For example :

Code : (element1 element2 (### [{] 0 [}]) )

In this example, when AI system interpret this code, the AI system take the element ‘element1’ as the temporary interpretation of the element ‘(### [{] 0 [}])’ . Thus the AI system will take the element (element1 element2 element1 ) as the temporary interpretation of the element ‘(element1 element2 (### [{] 0 [}]))’ . And then the AI system continue to interpret .

## 3.1.17/ Fam language specific strict instruction for interpreting : fam language specific default-accessing-by-multi-index interpreting rule .

The fam language specific code pattern : **([{] i1 i2 [}])**

The fam language specific code pattern : **([{] i1 i2 i3 [}])**

The fam language specific code pattern : **([{] i1 i2 i3 i4 [}])**

The fam language specific code pattern : **([{] i1 i2 i3 i4 i5 [}])**

And so on ..

These above fam language code pattern are used to specify fam language specific (default-assessing-by-multi-index) rule .

These rule are used to replace the fam language specific code pattern element by its specified nested element in the arbitrary specified level of its own hierarchy structure .

For example :

The fam language code pattern : **([{] i1 i2 [}])**

Where element ‘i1’ , element ‘i2’ are arbitrary element ; element ‘[{]’ and ‘[}]’ are fam language keyword element .

The specific code pattern element ‘([{] i1 i2 [}])’ always begin with parenthese ‘(‘ and end with parenthese ‘) , always have strictly 4 nested element at 1-level of its own hierarchy structure : ‘[{]‘ , ‘i1’ , ‘i2’ , ‘[}]’ with space between them .

Given element ‘i1’ , element ‘i2’ .

The fam language specific code pattern element : **([{] i1 i2 [}])**

The following illustrating equation can explain how AI system must use fam language specific instruction to interpret the fam language specific code pattern element ‘([{] i1 i2 [}])’ :

The illustrating equation : ([{] i1 i2 [}]) = (([{] i1 [}]) [{] i2 [}])

Thus in this case , the AI system firstly replace the element ‘([{] i1 i2 [}])’ by the element

‘(([{] i1 [}]) [{] i2 [}])’ . The element ‘(([{] i1 [}]) [{] i2 [}])’ is qualified fam language code pattern element of the fam language specific (default-accessing-by-single-index) rule . Thus the AI system next must apply fam language specific (default-accessing-by-single-index) rule to deal with the element ‘(([{] i1 [}]) [{] i2 [}])’ .

In the analogous way , there are such illustrating equation to explain how AI system must use fam language specific instruction to interpret element of the above fam language specific pattern :

([{] i1 i2 [}]) = (([{] i1 [}]) [{] i2 [}])

([{] i1 i2 i3 [}]) = (([{] i1 i2 [}]) [{] i3 [}])

([{] i1 i2 i3 i4 [}]) = (([{] i1 i2 i3 [}]) [{] i4 [}])

([{] i1 i2 i3 i4 i5 [}]) = (([{] i1 i2 i3 i4 [}]) [{] i5 [}])

([{] i1 i2 i3 i4 i5 i6 [}]) = (([{] i1 i2 i3 i4 i5 [}]) [{] i6 [}])

..

And so on ..

## 3.1.18/ Fam language specific strict instruction : fam language specific accessing-by-ID interpreting rule :

The fam language specific code pattern : **(TypicalElement [{{] IDelement Index [}}])**

Where element ‘typicalElement’ is arbitrary element ; element ‘IDelement ’ is arbitrary element; element ‘Index’ is arbitrary element ; ‘[{{]’ and ‘[}}]’ are fam language keyword element .

The specific code pattern element ‘(TypicalElement [{{] IDelement Index [}}])’ always begin with parenthese ‘(‘ and end with parenthese ‘)’ , always have strictly 5 nested element at 1-level of its own hierarchy structure : ‘typicalElement’ , ‘[{{]‘ , ‘IDelement’ , ‘Index’ , ‘[}}]’ with space between them .

Given element ‘typicalElement’ . Given element ‘IDelement’ . Given element ‘Index’ :

There is fam language specific code pattern element : ‘(TypicalElement [{{] IDelement Index [}}])’

The AI system interpret this specific code pattern element ‘(TypicalElement [{{] IDelement Index [}}])’ by the fam language specific (accessing-by-ID) rule :

When the AI system meet the such fam language specific code pattern element ‘(TypicalElement [{{] IDelement Index [}}])’ in the current fam language codepage , then the AI system firstly update the element ‘Index’ by applying other fam language specific rule (fam language specific replacement rule) to update : the element ‘Index’ -> the element ‘updatedIndex’ . If the AI system reckon that the element ‘updatedIndex’ is not integer , then the AI system replace the element ‘(TypicalElement [{{] IDelement Index [}}])’ by the element ‘()’ .

If the AI system reckon that the element ‘updatedIndex’ is integer , then the AI system firstly update the element ‘TypicalElement’ by applying other fam language specific rule (fam language specific replacement rule) to update : the element ‘TypicalElement’ -> the element ‘updatedTypicalElement’ . And if the AI system see that the element ‘updatedTypicalElement’ have no any 1-level nested element , then the AI system replace the element ‘(TypicalElement [{{] IDelement Index [}}])’ by the element ‘()’ . But if the AI system see that the element ‘updatedTypicalElement’ have 1-level nested element , then the AI system firstly update the element ‘IDelement’ by applying other fam language specific rule (fam language specific replacement rule) to update : the element ‘IDelement’ -> the element ‘updatedIDelement’ , and then the AI system iterate through all any-level nested element of the element ‘updatedTypicalElement’ to find the first result element satisfying condition : this result element contain 1-level nested element, which is absolute identical equal to the element ‘updatedIDelement’ and have 1-level index being equal to the integer ‘updatedIndex’ . Then the AI system replace the element ‘(TypicalElement [{{] IDelement Index [}}])’ by this result element . But if the AI system can not find any such result element , then the AI system replace the element ‘(TypicalElement [{{] IDelement Index [}}])’ by the element ‘()’ .

Thus the AI system update the fam language codepage .

## 3.1.19/ Fam language specific strict instruction : fam language specific default-accessing-by-ID interpreting rule :

The fam language specific code pattern : **([{{] IDelement Index [}}])**

Where the element ‘IDelement ’ is arbitrary element ; element ‘Index’ is arbitrary element ; ‘[{{]’ and ‘[}}]’ are fam language keyword element .

The specific code pattern element ‘([{{] IDelement Index [}}])’ always begin with parenthese ‘(‘ and end with parenthese ‘)’ , always have strictly 4 nested element at 1-level of its own hierarchy structure : ‘IDElement’ , ‘[{{]‘ , ‘Index’ , ‘[}}]’ with space between them .

Given element ‘IDElement’ . Given element ‘Index’ :

There is fam language specific code pattern element : ‘([{{] IDelement Index [}}])’

The AI system interpret this specific code pattern element ‘([{{] IDelement Index [}}])’ by the fam language specific (default-accessing-by-ID) rule :

## 3.1.20/ Fam language specific strict instruction : fam language specific index-by-ID interpreting rule :

The fam language specific code pattern : **(TypicalElement [{{] IDelement [}}])**

Where element ‘typicalElement’ is arbitrary element ; element ‘IDelement ’ is arbitrary element ; element ‘[{{]’ and ‘[}}]’ are fam language keyword element .

The specific code pattern element ‘(TypicalElement [{{] IDelement [}}])’ always begin with parenthese ‘(‘ and end with parenthese ‘)’ , always have strictly 4 nested element at 1-level of its own hierarchy structure : ‘typicalElement’ , ‘[{{]‘ , ‘IDelement’ , ‘[}}]’ with space between them .

Given element ‘typicalElement’ . Given element ‘IDelement’ :

There is fam language specific code pattern element : ‘(TypicalElement [{{] IDelement [}}])’

The AI system interpret this specific code pattern element ‘(TypicalElement [{{] IDelement [}}])’ by the fam language specific (index-by-ID) rule :

For example :

Code : (element1 (element2 (element3 element4)) element5)

In this example , the 1-level containing element of the element ‘element3’ is the element ‘(element3 element4)’ , the 2-level containing element of the element ‘element3’ is the element ‘(element2 (element3 element4))’ , the 3-level containing element of the element ‘element3’ is the element ‘(element1 (element2 (element3 element4)) element5)’ .

When the AI system meet the such fam language specific code pattern element ‘(TypicalElement [{{] IDelement [}}])’ in the current fam language codepage , then the AI system firstly update the element ‘TypicalElement’ by applying other fam language specific rule (fam language specific replacement rule) to update : the element ‘TypicalElement’ -> the element ‘updatedTypicalElement’.

There are 2 case : ‘Case 0’ and ‘Case 1’ as below :

Case 0 : If the AI see that the element ‘updatedTypicalElement’ have no any any-level nested element , then the AI system replace the element ‘(TypicalElement [{{] IDelement [}}])’ by the element ‘()’ .

Case 1 : if the AI system see that the element ‘updatedTypicalElement’ have nested element , then the AI system firstly update the element ‘IDelement’ by applying other fam language specific rule (fam language specific replacement rule) to update : the element ‘IDelement’ -> the element ‘updatedIDelement’ . Then the AI system will iterate all element , which are contained inside the element ‘updatedTypicalElement’ , to find the first result element , which is absolute identical equal to the element ‘updatedIDelement’ . If

the AI system can not find such result element , then the AI system replace the element ‘(TypicalElement [{{] IDelement [}}])’ by the element ‘()’ . If the AI system can find such result element , then the AI system replace the element ‘(TypicalElement [{{] IDelement [}}])’ by (the inherent index of this found result element inside the element ‘updatedTypicalElement’) .

Thus the AI system update the fam language codepage .

## 3.1.21/ Fam language specific strict instruction : fam language specific default-index-by-ID interpreting rule :

The fam language specific code pattern : **([{{] IDelement [}}])**

Where element ‘IDelement ’ is arbitrary element ; element ‘[{{]’ and ‘[}}]’ are fam language keyword element .

The specific code pattern element ‘([{{] IDelement [}}])’ always begin with parenthese ‘(‘ and end with parenthese ‘)’ , always have strictly 3 nested element at 1-level of its own hierarchy structure : ‘[{{]‘ , ‘IDelement’ , ‘[}}]’ with space between them .

Given element ‘IDelement’ :

There is fam language specific code pattern element : ‘([{{] IDelement [}}])’

The AI system interpret this specific code pattern element ‘([{{] IDelement [}}])’ by the fam language specific (default-index-by-ID) rule :

For example :

Code : (element1 (element2 (element3 element4)) element5)

In this example , the 1-level containing element of the element ‘element3’ is the element ‘(element3 element4)’ , the 2-level containing element of the element ‘element3’ is the element ‘(element2 (element3 element4))’ , the 3-level containing element of the element ‘element3’ is the element ‘(element1 (element2 (element3 element4)) element5)’ .

When the AI system meet the such fam language specific code pattern element ‘([{{] IDelement [}}])’ in the current fam language codepage , then the AI system firstly update the element ‘IDelement’ by applying other fam language specific rule (fam language specific replacement rule) to update : the element ‘IDelement’ -> the element ‘updatedIDelement’.

The AI system iterate through all element , which are contained inside the 1-level containing element of the element ‘([{{] IDelement [}}])’ , to find the first result element , which is absolute identical equal to the element ‘updatedIDelement’ . Thus it lead to totally 2 Case : ‘Case 1’ ‘Case 2’:

Case 1 : if in the above iterating process , the AI system can not find such result element inside the 1-level containing element of the element ‘([{{] IDelement [}}])’ , then the AI system replace the element ‘([{{] IDelement [}}])’ by the element ‘()’ .

Case 2 : if in the above iterating process , the AI system can find such result element inside the 1-level containing element of the element ‘([{{] IDelement [}}])’ , then the AI system replace the element ‘([{{] IDelement [}}])’ by (the inherent index of such result element inside the 1-level containing element of the element ‘([{{] IDelement [}}])’ )

Thus the AI system update the fam language codepage .

## 3.1.22/ Fam language specific strict instruction : fam language specific element-length interpreting rule :

The fam language specific code pattern : **([||] TypicalElement [||])**

Where element ‘TypicalElement’ is arbitrary element ; element ‘[||]’ and ‘[||]’ are fam language keyword element .

The specific code pattern element ‘([||] TypicalElement [||])’ always begin with parenthese ‘(‘ and end with parenthese ‘)’ , always have strictly 3 nested element at 1-level of its own hierarchy structure : ‘[||]‘ , ‘TypicalElement’ , ‘[||]’ with space between them .

Given element ‘TypicalElement’ :

There is fam language specific code pattern element : ‘([||] TypicalElement [||])’

The AI system interpret this specific code pattern element ‘([||] TypicalElement [||])’ by the fam language specific (element-length) rule :

When the AI system meet the such fam language specific code pattern element ‘([||] TypicalElement [||])’ in the current fam language codepage , then the AI system firstly update the element ‘TypicalElement’ by applying other fam language specific rule (fam language specific replacement rule) to update : the element ‘TypicalElement’ -> the element ‘updatedTypicalElement’.

Then the AI system replace the element ‘([||] TypicalElement [||])’ by the integer , which is equal to the number of all 1-level nested element of the element ‘updatedTypicalElement’ .

Thus this result integer can be 0 , 1 , 2 , .. so on .. .

## 3.1.23/ Fam language specific strict instruction : fam language specific default-element-length interpreting rule :

The fam language specific code pattern : **([||] [||])**

Where the element ‘[||]’ and ‘[||]’ are fam language keyword element .

The specific code pattern element ‘([||] [||])’ always begin with parenthese ‘(‘ and end with parenthese ‘)’ , always have strictly 2 nested element at 1-level of its own hierarchy structure : ‘[||]‘ , ‘[||]’ with space between them .

There is fam language specific code pattern element : ‘([||] [||])’

The AI system interpret this specific code pattern element ‘([||] [||])’ by the fam language specific (default-element-length) rule :

Lets use the temporary term ‘primary containing element’ to specify the element , which contain this fam language code pattern element ‘([||] [||])’ at the 1-level of the own hierarchy structure of this primary containing element .

When the AI system meet the such fam language specific code pattern element ‘([||] [||])’ in the current fam language codepage , then the AI system replace the element ‘([||] [||])’ by the result integer , which is equal to the number of all 1-level nested element of the element ‘primary containing element’ .

Thus this result integer can be 0 , 1 , 2 , .. so on .. .

## 3.1.24/ Fam language specific strict instruction : fam language specific referenced-element-length interpreting rule :

The fam language specific code pattern : **([|||] InherentIndex [|||])**

Where element ‘InherentIndex’ is arbitrary element ; element ‘[|||]’ and ‘[|||]’ are fam language keyword element .

The specific code pattern element ‘([|||] InherentIndex [|||])’ always begin with parenthese ‘(‘ and end with parenthese ‘)’ , always have strictly 3 nested element at 1-level of its own hierarchy structure : ‘[|||]‘ , ‘InherentIndex’ , ‘[|||]’ with space between them .

Given element ‘InherentIndex’ :

There is fam language specific code pattern element : ‘([|||] InherentIndex [|||])’

The AI system interpret this specific code pattern element ‘([|||] InherentIndex [|||])’ by the fam language specific (referenced-element-length) rule :

When the AI system meet the such fam language specific code pattern element ‘([|||] InherentIndex [|||])’ in the current fam language codepage , then the AI system firstly update the element ‘InherentIndex’ by applying other fam language specific rule (fam language specific replacement rule) to update : the element ‘InherentIndex’ -> the element ‘updatedInherentIndex’.

If the element ‘updatedInherentIndex’ is not an element from only 1-level nested integer element , then the AI system replace the element ‘([|||] InherentIndex [|||])’ by the element ‘()’ .

If the element ‘updatedInherentIndex’ is element from only 1-level nested integer , and the 1-level containing element of the element ‘([|||] InherentIndex [|||])’ do not have any contained element , which have (the inherent index in the 1-level containing element of the element ‘([|||] InherentIndex [|||])’) as the ‘updatedInherentIndex’ , then the AI system replace the element ‘([|||] InherentIndex [|||])’ by the element ‘()’ .

If the element ‘updatedInherentIndex’ is element from only 1-level nested integer , and the 1-level containing element of the element ‘([|||] InherentIndex [|||])’ have a contained element , which have (the inherent index in the 1-level containing element of the element ‘([|||] InherentIndex [|||])’) as the ‘updatedInherentIndex’ , then the AI system replace the element ‘([|||] InherentIndex [|||])’) by the integer number , which is equal to total 1-level nested element of this contained element .

For example :

Code : (A B 15 (1 (x (y y) ‘ z) “) ([||| (3 1) [|||]) 5)

Result must be : (A B 15 (1 (x (y y) ‘ z) “) 4 5)

## 3.1.25/ Fam language specific strict instruction : the procedure of executing fam language specific instruction :

The first stage of fam language codepage interpretation is the stage , in which AI system apply only fam language specific strict instruction to : replace directly each element in fam language codepage by other specified element , and to add or modify new user defined fam language specific element-replacement rule to the fam language codepage rule repository .

In the stage of fam language specific instruction realizing , the AI system apply fam language specific instruction to serially interpret element by element : AI system firstly interpret element with lower logical executing order , then AI system go to interpret element with higher logical executing order . Logical executing order of the element is the (logical executing order of (this element inherent index)) , for example the logical executing order of (the relative universal inherent index of this element) in the fam language codepage .

The most distinct principle of fam language specific strict instruction is that it continuous replacement of element directly on the fam language codepage . Thus the fam language codepage evolve , and act as independent memory for all element and operation . Fam language specific rule is stored in the fam language codepage rule repository . Beside fam language codepage and fam language codepage rule repository , there is no need of other memory in the stage of realizing fam language specific strict instruction . Thus the stage of fam language specific instruction realizing do not require any kind of AI system memory , and do not depend on any AI system specific memory management mechanism . The stage of fam language specific instruction realizing use the fam language codepage rule repository to store fam language specific rule , and use the continuously updated fam language codepage as memory to store updated element thank to instantly replace element directly at the element place by other specified updated element .

**3.1.25.1/ Theorem about the immediate next queueing element in the stage of fam language specific instruction realizing** :

Suppose that the AI system already interpreted the element ‘A’ .

If the interpreted element ‘A’ have 1-level nested element , then the immediate next queueing element for interpreting after the interpreted element ‘A’ is always the first 1-level nested element of the interpreted element ‘A’ .

If the interpreted element ‘A’ do not have 1-level nested element , and is not the last 1-level nested element inside its 1-level containing element , then the immediate next queueing element for interpreting after the interpreted element A is the next 1-level nested element (which stay immediately in-the-right after the element ‘A’) of the 1-level containing element of the interpreted element ‘A’ .

If the interpreted element ‘A’ do not have 1-level nested element , and the interpreted element ‘A’ is the lasted 1-level nested element in its 1-level containing element , then the immediate next queuing element for interpreting after the interpreted ‘A’ is the element , which stay immediate right after the 1-level containing element of the interpreted element ‘A’ in the 2-level containing element of the interpreted element ‘A’ .

For example :

Code : ((XX (B C D) 123 BCD (3 2 1)) PQL)

In the above example , suppose that the AI system just already interpreted the element ‘(B C D)’ as a result ‘(B C D)’ . Thus by the above theorem of immediate next queuing element for interpreting , the

immediate next queueing element for interpreting after the interpreted element ‘(B C D)’ is the element ‘B’ .

In the above example , suppose that the AI system just already interpreted the element ‘BCD’ as a result ‘BCD’ . We see that the element ‘BCD’ do not have 1-level nested element . Thus by the above theorem of immediate next queuing element for interpreting , the immediate next queueing element for interpreting after the interpreted element ‘BCD’ is the element ‘(3 2 1)’ .

In the above example , suppose that the AI system just already interpreted the element ‘(3 2 1)’ as a result ‘(3 2 1)’ . The immediate next queuing element after the ‘(3 2 1)’ is the element ‘3’ . And the immediate next queuing element after the element ‘3’ is the element ‘2’ . And the immediate next queuing element after the element ‘2’ is the element ‘1’ . And the immediate next queuing element after the element ‘1’ is the element ‘(3 2 1)’ . Thus the element ‘(3 2 1)’ is already unchanged under repeated applying all fam language replacement rule to this element ‘(3 2 1)’ and to all its contained element . After that , because the unchanged interpreted element ‘(3 2 1)’ is the lasted 1-level nested element in its 1-level containing element , thus the immediate next queuing element after the interpreted unchanged element ‘(3 2 1)’ is the element ‘PQL’ .

Apply the above theorem , there is practical procedure instant in-place replace element in fam language codepage :

**3.1.25.2/ Procedure of interpreting element by fam language specific instruction :**

Step 1 : access element , for example the element ‘AA’ . Firstly apply fam language specific replacement rule to replace the whole element by other specified element . This replacement is instantaneous and in-place, directly modifying the Fam language codepage at that specific position .

In this process , some new rule can be added to the fam language code pattern rule-declaring element . Until the element ‘AA’ become unchanged under applying all saved fam language specific replacement rule .

Step 2 :

Crucially, after an element undergoes such an instant in-place replacement, the AI system must immediately and continuously focus its serial logical executing order on the newly inserted content that now occupies that position. It will persistently apply all applicable fam language specific replacement rules to this newly inserted content (and its continuously evolving nested elements) in their serial logical executing order , without interruption , until this newly inserted content , as a whole , becomes unchanged under all saved fam language specific replacement rules.

Only after this complete internal stabilization of the replaced element through continuous in-place replacement , will the AI system proceed to determine the immediate next queueing element for interpreting in the broader context of the codepage , by strictly following the rules detailed in Theorem (3.1.25.1/).

**3.1.25.3/ Approximating principle :**

If human user or AI user write complexed mutual related fam language specific code pattern rule-declaring element in unappropriate way , then fam language specific instruction process can lead to unconverged fam language codepage , which will become infinite long and not converged .

There are also scenario , in which the instant in-place replacement process of an element can not converge to unchanged resulted element .

In such case , the AI system can decide to stop after some appropriate number of task cycle , and take the instant updated fam language element and fam language codepage as temporary conceptual interpretation of the original fam language element and original fam language codepage , to go to next stage of interpreting by AI system specific model .

For example :

Code : (A [=] (A +1)) A

In the above example , it lead to unconverged result and infinite long element :

-> (A [=] (A +1)) (A + 1)

-> (A [=] (A +1)) ((A + 1) + 1)

-> (A [=] (A +1)) (((A + 1) + 1) + 1)

-> (A [=] (A +1)) ((((A + 1) + 1) + 1) + 1)

.. so on ..

In this case , the AI system can decide not to continue forever to generate infinite long fam language codepage , but the AI system stop process at some point , and take the latest result '(A [=] (A +1)) ((((A + 1) + 1) + 1) + 1)' as the temporary conceptual interpretation of the original fam language codepage .

## 3.2/ CALCULATION THREADING REGIME :

Fam language specify 2 default choice of calculation threading regime for AI system to parse fam language element :

### 3.2.1/ The SERIAL CALCULATION THREADING REGIME :

By choosing this serial calculation threading regime to do calculating thread, the AI system will serially do these thread.

Fam language emphasize that when parsing fam language code page, the AI system must in SERIAL CALCULATION THREADING REGIME to parse (interpret) all element in 1-level of the hierarchy structure of fam language code page.

### 3.2.2/ The SYNCHRONISED EVOLUTION CALCULATION THREADING REGIME :

By choosing this synchronized evolution calculation threading regime, the AI system do these thread in the following procedure :

+ The AI system divide time into small time interval, for example timeInterval1, timeInterval2, ..,timeIntervalN .

+ In each of these small time interval, the AI system will parallelly simultaneously ASYNCHRONOUSLY do all these thread.

+ At the every end moment of each these time interval, the AI system will make very small pause of all these parallel asynchronous thread, so that each of these thread can take updated information from other thread to apply in it.

+ Then, in the next small time interval, the AI system will parallelly simultaneously ASYNCHRONOUSLY continue doing all these thread.

Fam language emphasize that when parsing fam language code page, the AI system must IN SYNCHRONISED EVOLUTION CALCULATION THREADING REGIME to parse (interpret) all element in 2-level and in higher level of the hierarchy structure of fam language code page.

## 3.3/ Instruction for AI system to choose calculation threading regime :

There are 2 distinct stage of interpreting : The first stage of applying only fam language specific instruction (rule) and with highest priority for fam language specific instruction ; and the second stage of applying only AI system specific rule .

In first stage of realizing fam language specific instruction , the AI system follow only fam language specific instruction (fam language specific replacement rule) , to interpret (evolve) the original fam language codepage into some final updated fam language codepage , which is the temporary conceptual interpretation of the original fam language codepage . In this stage , choose serial calculation threading regime , to firstly interpret element with lower element logical executing order , then go to interpret element with higher element logical executing order .

In the second stage of realizing AI system specific interpreting instruction : after the first stage of interpreting , the AI system receive the updated fam language codepage . In second stage , the AI system will freely apply only their own AI specific interpreting rule without any required counting of fam language specific instruction , to interpret this updated fam language codepage into final interpretation . But fam language emphasize that the AI system should apply conceptual synchronized evolution calculation threading regime in this second stage .

**4/ Contact :**

Contact : fam language project :

famlanguage@gmail.com

github.com/famtyen/famlanguage

youtube.com/@famlanguage

youtube.com/famlanguage