International Journal of Hybrid Information Technology
Vol. 10, No.2 (2017), pp. 163-176
http://dx.doi.org/10.14257/ijhit.2017.10.2.16

The Quantification of Seal Calculus in the Software System

Nan Zhu

Xinxiang University.Henan.China
Corresponding E-mail:Zhunn@163.com

Abstract

Seal calculus is a kind of tool targeting at the modeling of the mobile system. Since there are some restrictions on the regions and levels, the seal calculus, when compared with π calculus, could imitate the actual concurrent mobile system in a more convenient and efficient way. However, in the realistic system modeling analysis, the analyses concern not only about the internal business logic of the system, as well as the restrictions on the region and the levels processed in the analysis, but also about whether the system could satisfy the resource demands of the process . In order to imitate as closely to the actual modeling as possible, this paper introduces the concept of quantification analysis to carry on nine kinds of extensions in the use of seal calculus. Besides the preservation of the business logic analysis within the system, the paper analyzes emphatically on how the external resource supplies limit the resource demands of the process, in order to realize the quantitative and qualitative analysis of the model.

Keywords: *seal calculus; quantification of seal calculus; π calculus; mobile modeling*

1. Introduction

In the design and development of software system, it is crucial to establish a scientific software model. Since the software system nowadays requires concurrency, mobility and distributiveness, the seal calculus has become one of the most appropriate choices to carry on the system modeling.

Advanced by Castagna and Vitek in 1998 as a kind of process algebra, the seal calculus[2,3,4,5] is a form of distributional process calculus which describes the features of mobility and security . That is, it is just a variant of π-calculus [11,12,13,14]. But in comparison with π-calculus, the seal calculus evolves a lot by adding to the conception of link mobility the idea of regions and levels similar to ambient calculus [6,7,8,9,10]. In other words, though the seal calculus keeps the concepts of names and the ways of using channels for interaction like π-calculus does, it turns a flat space into a space with different regions and levels, with additional bonus of increasing the security mechanism of seal communication and mobility. This feature could facilitate the designer to describe the actual concurrent mobile system to a further degree.

But we should have sober awareness that the seal calculus also has its own limitations during the modeling process in many reality systems. Because the seal calculus itself could only describes the business logic of the system itself in a qualitative way, this causes the problem of being unable to analyze quantitatively the resource demands of the system itself and the resource supplies outside the system, when the modeling is carried on in seal calculus. The consequence of this is that the designer could do modeling only when he assumes that all the external resources, such as computation resources, memory resources and network resources etc. are in an ideal state of infinite supplies, and without the necessity to consider the time of taking up the resources in the process. But as we know, all the processes actually have the resources demands and all the external resource supplies are finite. All the related resources must be requested before the execution of the process. Only when the resources meet the needs of the process could the process runs normally and

ISSN: 1738-9968 IJHIT

properly. Of course, the time of taking up the resources are surely limited in the process. The lack of targeting the resource demands within the system and the resource supplies outside the system has become an inherent deficiency of seal calculus for quantitative evaluation, which will cause potential hazards, especially in the aspect of stability, when the seal calculus is applied to do modeling and developing a software system.

In view of the limitation of the seal calculus, this paper will deal with the problem by adding to the seal a constraint function vector for the external resources, and by increasing a matrix function for the resource demands of the process itself. Only when the various resource components of the demand functions in the process itself are fewer than the corresponding components of the constraint functions of resources in the seal, could the process runs normally. What's more, in order to make better quantification for the resources supplies and related consumption, and in order to make more convenient simulation of the realistic situation, this paper extends part of the syntax and the semantics in the seal calculus to form a device which can analyze modeling both qualitatively and quantitatively --- the quantification of the seal calculus.

A BRIEF DESCRIPTION OF SEAL CALCULUS

The syntax of the seal calculus is as follows[1]:

Names：	Locations：	Actions:	Processes:
a,b,……x,y,z,……∈ Name	η = *	a::= $\bar{x}^{\eta}(y_1,\ldots,y_n)$ output	P,Q,R ::= 0 inactivity
	local ↑ up	$x^{\eta}(y_1,\ldots,y_n)$ input	P|Q concurrent execution
	z down	$\bar{x}^{\eta}\{y\}$ send	(V*x*)P restriction
		$x^{\eta}\{y_1,\ldots,y_n\}$ receive for n >= 0	a.P prefixing
			n[P] seal
			!a.P replication

Assumingη_1 andη_2 are two locations, and y is a name, then we can define

Share channels: $synch_y^{S}(\eta_1,\eta_2) = (\eta_1=y \wedge \eta_2 =\uparrow)$

Located channels: $synch_y^{L}(\eta_1,\eta_2) = (\eta_1=y \wedge \eta_2 =_*) \vee (\eta_1=_* \wedge \eta_2 =\uparrow)$

Deductive rules for the seal calculus

Local writing $x^*(\vec{u}).P \mid \bar{x}^*(\vec{v}).Q \rightarrow P[\vec{v}/\vec{u}] \mid Q$

Write in $\bar{x}^{\eta 1}(\vec{w}).P \mid y[(v\vec{z})(x^{\eta 2}(\vec{u}).Q_1 \mid Q_2)] \rightarrow P \mid y[(v\vec{z})(Q_1[\vec{w}/\vec{u}] \mid Q_2)]$

Write out $x^{\eta 1}(\vec{u}).P \mid y[(v\vec{z})(\bar{x}^{\eta 2}(\vec{v}).Q_1 \mid Q_2)] \rightarrow (v\vec{v} \cap \vec{z})(P[\vec{v}/\vec{u}] \mid y[(v\vec{z} \setminus \vec{v})(Q_1 \mid Q_2)])$

Local movement $x^*\{\vec{u}\}.P_1 \mid \bar{x}^*\{v\}.P_2 \mid v[Q] \rightarrow P_1 \mid u_1[Q] \mid \ldots \mid u_n[Q] \mid P_2$

Move in $\bar{x}^{\eta 1}\{v\}.P \mid v[S] \mid y[(v\vec{z})(x^{\eta 2}\{\vec{u}\}.Q_1 \mid Q_2)] \rightarrow P \mid y[(v\vec{z})(Q_1 \mid Q_2 \mid u_1[S] \mid \ldots \mid u_n[S])]$

Move out

$x^{\eta 1}\{\vec{u}\}.P \mid y[(v\vec{z})(x^{\eta 2}\{\vec{v}\}.Q_1 \mid v[R] \mid Q_2)] \rightarrow P \mid (vfn(R) \cap \vec{z})(u_1[R] \mid \ldots \mid u_n[R] \mid y[(v\vec{z} \setminus fn(R))(Q_1 \mid Q_2)])$

2. The Quantification of Seal Calculus

A better imitation of the reality system could be achieved by the quantification of the seal calculus. In this paper, based on the traditional seal calculus, the following nine kinds of extensions are advocated in the quantification of the seal calculus.

The Extension on the Meanings of Seal Labels

In the quantification of seal calculus, seal labels are no longer just simple ones to identify seal names. Since every seal has its own function vector of resource supplies, then on the base of the traditional seal calculus, a seal label could be extended to be an ordinal pair of a label and its function vector, that is, to extend n in n[p] to be <n F_n>. In the expression, the meaning of n to be a label identifying a seal is still unchanged, while F_n is the function vector of resources supplies for the seal. And in the function vector, each function component represents a change of resource supplies.

In most cases, the system will adopt the time to be the variable of this function. If indeed the system uses the time as the variable for the resource to provide the function, the processes of the seal could run safely only when the total sum of the resource requirements within each time slice of each process is less than the function components provided by resources corresponded to the seal. What is worth mentioning is that the amount of resource supplies has nothing to do with the level where the seal lies, which means that the resource provided by the child seal could be in greater amount than that by the parent seal.

In the actual system design, the seal generally saves a certain amount of the resource-and-time redundancy according to what is actually happening, in order to guarantee that the system could run in a more stable way. Under the condition when the ambiguity is unlikely to happen, the ordinal pair identifying the seal could be simplified to just one name. For example, <n F_n>[P] could be simplified to n[p]. Although n[p] is still consistent in the written form with that in the traditional seal calculus, it is semantically different from the latter, because the concept of resource supplies is added to it.

The Corresponding Extension of the Meaning of the Channel Name in the Process

In π calculus and the traditional seal calculus, the process, as a recursive concept, is a complex combination of the name and another process. Following π calculus and the traditional seal calculus, the quantification of the seal calculus keeps the syntactical routine by also using the capital letter to express the process and the lowercase letter to express the name. Still similar to π calculus and the traditional seal calculus, the letter combination is also adopted to express processes and names so that the system description could be more straightforward and explicit in reality.

In the quantification of the seal calculus, the meaning of the name which serves as the channel is extended to be the ordinal pair: "the name, the function matrix". In other words, suppose a process is expressed as P=a.P1 in the traditional seal calculus, it will be extended to be P=<a,X_a>.P1 in the quantification of seal calculation. In the ordinal pair, a is just a name, but X_a means one step carried on by P, a function matrix of resource demands before turning into P1. The row vector for the matrix of resource demands in details is the serial number for the seal, while its column vector is the serial number of the resource types. For example, the element f_{ij} in the demand matrix means the function f of the resource demand in the jth process of the ith seal.

The reason why matrix functions are adopted in the ordinal pair is that in the actual uses, some process may not only occupy the resource of its own seal, but also take those of other seals. Therefore, as far as the process is concerned, the resource types and the resource location need to be explained clearly, which requires the matrixes to describe

how well the resources are used in the process. Under the condition when the ambiguity is unlikely to happen, the name that serves as a channel and the corresponding ordinal pair of the function matrix of the resource demands could be simplified into just one name. For example, <a,X_a>. P1 could be simplified into a.P1. Similar to what has been discussed in Section 3.1, the meaning in the quantification of the seal calculus is quite different from that in the traditional seal calculus, because the concept of resource demands is added to it.

In the quantification of the seal calculus, it is generally believed that once starting up, the process will request resources for running according to its own fixed resource function. Of course in certain cases, say, if some process has no specific limitation to the time, but only the requirements for the total amount of the assignment tackled and the cut-off time, then the function for resource use could make self-adjustment according to the real-time situation provided by the resource in the seal. Therefore, the conceptual introduction of both resource demands of the process and the restriction of the external resources are one of the most important differences between the seal calculus and the quantification of the seal calculus.

For instance, in the seal calculus $P3|y[\overline{a}^*.P1|a^*.P2] \to P3|y[P1|P2]$, but in the quantification of the seal calculus, if the seal y could not provide the resources of x reaction, then the reaction would be unable to carry on. In other words, $P3|<y,F_y>[<\overline{a}^*,X_{a1}>.P1|<a^*,X_{a2}>.P2] \not\to P3|<y,F_y>[P1|P2]$

Under the condition when ambiguity is not likely to happen, in order to write conveniently, the expression above could be written as $P3|y[\overline{a}^*.P1|a^*.P2] \not\to P3|y[P1|P2]$.

By building up a simple system, we may have a thorough understanding of the differences between the traditional seal calculus and the quantification of seal calculus through their comparison.

Assuming a system is named the concurrent service system (CSS in short). Located in n different client areas of the CSS system, n concurrent client processes will apply for services to the control process of the service areas. The service processes can concurrently offer services to different client processes which have applied for the service.

In the traditional seal calculus, this could be expressed as follows.

In the n concurrent client processes, suppose [0..n-1] =N,

$$\prod_{i\in N} client_i[open_{\uparrow} apply \mid \overline{apply}(mission_i\ client_i).\overline{mission_i}] \quad (1)$$

then the service process is: $!apply(m\ c).(open_c\ \overline{m} \mid m)$ (2)

The whole system formed is:

$$!apply(m\ c).(open_c\ \overline{m} \mid m) \mid \prod_{i\in N} client_i[open_{\uparrow} apply \mid \overline{apply}(mission_i\ client_i).\overline{mission_i}] \quad (3)$$

Suppose that there are m client processes (0=<m<=n-1) which successfully propose the task application, and suppose the set of the serial numbers of the client processes belong to the set M, then the system will become:

$$!apply(m\ c).(open_c\ \overline{m} \mid m) \mid \prod_{j\in M} (open_{client_j}\ \overline{mission_j} \mid mission_j) \mid \prod_{k\in M} client_k[\overline{mission_k}] \mid \prod_{i\in N-M} client_i[open_{\uparrow} apply \mid \overline{apply}(mission_i\ client_i).\overline{mission_i}] \quad (4)$$

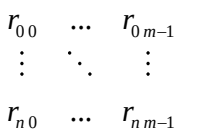
In this way, after the deduction from the description of the system by the traditional seal calculus, the process in the system ends up running normally and concurrently without any restriction. So eventually Expression 4 will become:

$$!apply(m\ c).(open_c\ \overline{m} \mid m) \mid \prod_{i\in N-M} client_i[open_{\uparrow} apply \mid \overline{apply}(mission_i\ client_i).\overline{mission_i}] \quad (5)$$

However, many unexpected errors may often happen in the actual system running. One of the important reasons lies in the lack of considering the spatio-temporal limits of the resources in the running environment of the system by the designer who, in many cases, takes into consideration only the business logic of the system itself when modeling. As far as Expression 4 is concerned, if m missions are carried out at the same time, and if the system cannot provide the resources needed in the running of the missions at a certain point of time, then the system is likely to err and probably unable to become Expression 5. Therefore, in order to avoid such problems and make more accurate analyses of the system, a further quantification is necessary on the base of the traditional seal calculus.

The simplest quantification is actually the quantification towards the constants of the resources demanded by the process, which means the resource demands by the process must be a constant in the entire running cycle of the process. Since what is needed is a constant, in order to guarantee the running security of the system, generally we have to choose the maximum value needed for the resource by the process at a certain point of time.

Suppose the entire system has provided m resources altogether, R_0 … R_{m-1} respectively, and suppose the seals in the system are CSS, $client_0 client_1 \ldots client_{n-1}$, respectively, then the vector provided by the seal resources is the one that contains m elements, while the matrix for the resource demands of the process is the one with m*(n+1). In the matrix of the resource demands, the column vector is the resource type and the row vector is the seal. It is just like figure 1



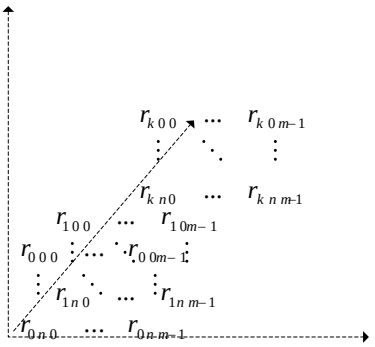


Figure 1 **Figure 2**

So the matrix r_{ij} means that the demand to the jth type of resource is r in the ith seal. Then considering the resource demands of k process missions in CSS, the figure is as figure 2.

From figure 2, it can be seen that the two-dimensional matrix of resource demands by every process is synthesized into a three-dimensional matrix of resource demands for a process set. In this way, suppose there is a value i, it will make $R_j < \sum_{i=0}^{k} r_{inj}$. This means that at this moment the resource needed by the process has exceeded that provided by the seal, which may probably cause errors to happen. Therefore, in CSS, when the amount of resource demands by the process has been close or reached that of the resource supplies of CSS, some control should be made to the system. At this point, the new process is not allowed to start running until some process releases part of the resources in the seal, so that the resource provided in CSS could satisfy the needs for the new process.

In this calculation, the resource needed by the process is the maximum amount of resource needed by the process at a certain point of time. But the quantification of the constants, as a disadvantage, cannot offer high efficiency of the system. That's why the time quantification has to be necessarily introduced to improve the system efficiency. Thus, in the matrix of resources demands, each demand constant needs to be changed into

the resource function f(t) which uses the time as the variable. The seal responsible for providing resources is also a function F(t) which changes along with the time. Suppose the seal provides m resources: $F_0..F_{m-1}$ respectively, then the matrix of resource demands as figure 3:

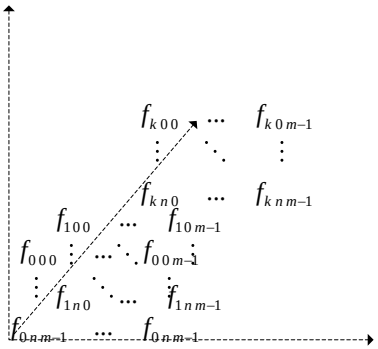


Figure 3

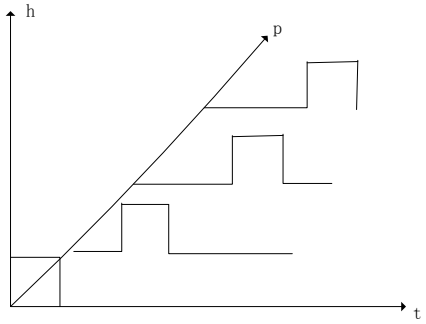


Figure 4

From figure 3, we could see that only when $F_j < \sum_{i=0}^{k} f_{inj}$ happens could the system have the possibility to err.

This could be illustrated from an example. Under the condition that the function provided by the seal to the resource h is a constant with the value 1, suppose what are waiting for execution are 4 processes [p0..p3] which, in running, only make demands for the resource h, suppose the first quantification method mentioned in 3.1 is adopted with the use of the resource h by each process for 1 time, then each time at most only one process will be carried out. However, if the second quantification method is adopted with the additional time quantification of classifying the time accordingly into, 4 slices [t0..t3], in which p0 could only use the resource in t0, similarly, p1 in t1, p2 in t2, and p3 in t3, then the four processes could run concurrently without necessarily waiting for the previous process to finish. So the efficiency could be greatly improved. This can be well presented in figure 4.

What needs to be noticed is that some resource demands may be those not requested by the process, their goal is achieved if they finish the missions as requested in the given time. In this case, the system needs to make an overall arrangement according to the actual situation.

3. Flexible Permission of Seal Resources to have Some Temporary Changes

The resources of the reality system could change dynamically. Generally speaking, there are two kinds of resource changes. The one is the planned change which can be described by the use of functions. The other is the temporary change. This kind of change may be caused by many different situations: the increase or the decrease of a server in some region; the increase or the decrease of computing resources and storage resources or peripheral resources; the temporary renting of the network resources from network carriers; the temporary addition or loss of network resources caused by the positional change of aeronautical communication or satellite communication--- all of these may lead to the change of resources. In the quantification of seal calculus, the computable feature of the resources can simulate this kind of situation.

Alter resources:

$$\overline{x}^{*} < y, fun2 > .P1 \mid x^{*} < y > .P2 \mid < y, fun1 > [P3] \rightarrow P1 \mid P2 \mid < y, fun1 + fun2 > [P3]$$

The channel x here indicates clearly the seal label y and the resource information fun2 in the delivery (It is noteworthy that, in fun2, not all resource components are necessarily positive, because the change of the resources may go through either the increase or the decrease).

3.1 Permission of the Process Movement

The traditional seal calculus only allows for the process interactions and seal movements, the system configuration and resource scheduling may suffer in efficiency. In order to solve the problem and better imitate these two aspects in the reality system, the concept of the process movement is introduced in the quantification of the seal calculus. In other words, the process is endowed with the ability to be moved in or out of the seal.

In the traditional seal calculus, there has already been the moving mechanism of regions and levels as well as the link moving mechanism. Although the process movement could be roughly imitated by the link movements in the traditional seal calculus, the concept of the process movement still needs to be unambiguously introduced in the quantification of seal calculus. In the traditional seal calculus, the process movement imitated through the delivery of names is a kind of active movement, while in the quantification of the seal calculus, the process movement is a passive one as far as the process is concerned. The passive way greatly facilitates the real-time dispatch to the system. When the system resources go through some dynamic changes, probably due to the dynamic increase or decrease of the network resources, the computing resources or the storage resources, the method of seal movement could be adopted for the simulation of the dynamic changes of the resources by moving all the processes in the original seal to the seal with new resource configuration.

In the detailed discussion of the reality system, there are two kinds of process movements. The first is having a backup process in the destination seal, so that when the process makes some movements, the system only needs to send a message to start the backup process. The other kind is moving all the process data to the destination seal. Both situations can be well expressed in the quantification of the seal calculus, but they have requirements for different amount of resources and different uses of the time. For the convenience of the expression, the content within {} is specified to be the seal, while that within () is specified to be the process.

process move in and seal create local

$$x^*\{< y, fun >\}.P1 \mid \overline{x}^*(Q).P2 \mid < y, fun > [P3] \mid Q \to P1 \mid < y, fun > [P3 \mid Q] \mid P2$$

When there is no seal labeled as y, the system has to create one:

$$x^*\{< y, fun >\}.P1 \mid \overline{x}^*(Q).P2 \mid Q \to P1 \mid < y, fun > [Q] \mid P2$$

The expression shows moving the process to the seal with the function fun provided by the newly created resource y.

In the quantification of the seal calculus, the operation of creating a new seal is a well-considered supplement for the original operations of seal duplication and seal destruction. The rule addition of "creating a new seal" perfects the seal operation in the quantification of the seal calculus.

Among them, the original seal duplication[1] is: (copy x as z).P = $(vy)(\overline{y}^*\{x\} \mid y^*\{x, z\}.p).$

The original seal destruction [1]is: (destroy x).P = $(vy)(\overline{y}^*\{x\} \mid y^*\{\}.p).$

process move out :

$$x^y\{*, fun\}.P1 \mid < y, fun1 > [\overline{x}^{\uparrow}(v).Q \mid v \mid P2] \to P1 \mid v \mid < y, fun1 > [Q \mid P2]$$

In the expression, fun is the function vector provided by the resource of the seal at the uttermost level.

3.2 Permission of the Process to Lock and Unlock

In the actual system design, there exists some possibility of fixing a certain process in a specific environment for execution without any movement at will. For instance, some decoding process may be required to work only in some specific area, while some video playback process may be required to carry out in a specified area, say, a conference room. That is to say, if no permission mechanism is designed for the movement operation, it is liable for such mistakes like "wrong movement" to appear. Therefore, in order to increase the security of the processes and the seal movements, two kinds of operations are necessary to be added in the quantification of the seal calculus: the positional locking and unlocking.

$\overline{x}^{*} < \nabla, P3 > .P1 | x^{*} < \perp, P3 > .P2 | P3 \rightarrow P1 | P2 | \nabla P3$ Positional locking

$\overline{x}^{*} < \Delta, P3 > .P1 | x^{*} < \perp, P3 > .P2 | \nabla P3 \rightarrow P1 | P2 | \Delta P3$ Positional unlocking

In the expression, $\perp$:stands for the attribute of receiving the information of the process position, ∇: stands for the position of the process being fixed without any ability to move, and Δ stands for the position of the process having the ability to move freely. But in the situation where no misunderstanding arises, Δ can be omitted. In other words, $\overline{x}^{*} < \Delta, P3 > .P1 | x^{*} < \perp, P3 > .P2 | \nabla P3 \rightarrow P1 | P2 | P3$.

What needs to be noted is that the process is only related to the seal, which means, when the process is fixed in some seal, the movement of the seal itself will not be influenced by the action of the process immobilization.

3.3 Permission of Seal Names and Superscript Locations in the Process to be Revised

In the actual dispatch of the system process, the designer usually allocates the use of resources within a region through the movements of processes. However, the change in the region and the level of the process may lead to the communication failures of some channels. Therefore, before the movement of the process, a full consideration of other processes related to this one is crucial. What's more, after the process movement, sometimes some revisions have to be made to the superscript of the process (in the seal where the process is after the movement, or in the seal which the process is going to communicate with. So, in comparison with the traditional seal calculus, the revisions in seal names and superscript processes are believed to provide a better support for the seal movements and process movements in the quantification of the seal calculus.

The change of the seal names could be done through moving the process in the original seal to a new seal and then moving that seal back. In order to simplify the operation, we also introduce the name changing operation for the seal, by using # to complete the name changing operation.

$\overline{x}^{*} < \#, z > .P1 | x^{*} < \#, y > .P2 | < y, fun1 > [P3] \rightarrow P1 | P2 | < z, fun1 > [P3]$

In the expression, y and z can make the vector generalization.

The symbol @ can be used to identify the name changing operation for the locations.

$\overline{x}^{*} < @y, \eta 2 > .P1 | x^{*} < @y, \eta 1 > .P2 | y^{\eta 1}.P3 \rightarrow P1 | P2 | y^{\eta 2}.P3$

A simple example can illustrate this point.

main= $P1 | < y, fy > [P2 | \nabla P3 | \nabla P4 | \nabla P5]$,In this expression,P1 $= < a^{y}, FP1 >$, P2= $< \overline{a}^{\uparrow}, FP2 >$

Supposition 1: In a system named 'main', the resource vector whose function is providing resources is named 'fmain'. Supposition 2: The resource vector of the seal labeled y is insufficient to provide the resources for the simultaneous running of $P2 | \nabla P3 | \nabla P4 | \nabla P5$. Supposition 3: The system P2 is a process with the permission to

move, which means that the P2 result and its form of expression will not be influenced by the calculation location. But the rest of processes (P3, P4 and P5) have to be carried out only in the seal y. Supposition 4: The resource vector fmain has enough resources only for the normal running of P1 and P2 (not the rest). Therefore, in order to guarantee the normal running of all the processes in the system, an appropriate scheduling or dispatch is absolutely necessary for the processes of the system.

There are two steps for the scheduling or dispatch. The first one is moving P2 out of the seal y. The second step is revising the superscript of the channel a, and moving the process M (which plays the part of moving) to the seal y, in which M=$<\bar{v}^{\uparrow}\{P2\}, FM>$. In this system, since the process M only plays the part of moving, under the condition that its occupation of the resource is not too much, the seal y could hold $P2|\nabla P3|\nabla P4|\nabla P5|M$.

$$x^{*}\{<y, fy>\}|\bar{x}^{*}\{M\}|M|P1|<y, fy>[P2|\nabla P3|\nabla P4|\nabla P5]\rightarrow P1|<y, fy>[P2|\nabla P3|\nabla P4|\nabla P5|M]$$

Then the movement of P2 is carried out. That is, the movement of P2 out of the seal y is completed.

$$v^{y}\{*, fun\}|P1|<y, fy>[\bar{v}^{\uparrow}\{P2\}|P2|\nabla P3|\nabla P4|\nabla P5]\rightarrow P1|P2|<y, fy>[P2|\nabla P3|\nabla P4|\nabla P5]$$

However P1=$<a^{y}, FP1>$ P2=$<\bar{a}^{\uparrow}, FP2>$,which cannot respond in the system 'main', leading to the necessity to revise their superscripts.

$$\bar{u}^{*}<@a,*>|u^{*}<@a,y>|\bar{w}^{*}<@a,*>|w^{*}<@a,\uparrow>|<a^{y}, FP1>|<\bar{a}^{\uparrow}, FP2>|<y, fy>[\nabla P3|\nabla P4|\nabla P5]\rightarrow$$
$$<a^{*}, FP1'>|<\bar{a}^{*}, FP2'>|<y, fy>[\nabla P3|\nabla P4|\nabla P5]$$

In this way, the system eventually turns out to be $<y, fy>[\nabla P3|\nabla P4|\nabla P5]$. Still, one has to notice that, after the process being moved, the function matrix of the resource demands of the process will also have some corresponding changes, which means that the location where the resource is applied for, the seal, will also have some corresponding changes.

3.4 The restriction of Operators which have the Ability to Make Infinite Suplication

In order to make more accurate estimation of how much the resource is used, the duplication of the process is needed to make quantified extension in the quantification of the seal calculus. $!^{n}$P, means that at most n P processes are permitted to move concurrently. So when the number of the P processes reaches n in the seal, other P processes are supposed to stop starting up. Only when one or more than one P processes die away and the total number of P processes in the seal is fewer than n can other P processes move in to fill up.

3.5 Permission to Revise the Restriction of Duplication Operators

The revision of the restriction on duplication operators means the revision of the superscript !, in which the increase or the decrease of the superscript numbers indicates that of the number for concurrent processes. Attention must be paid to the situation of the extra processes in the previous movement, because, when the restriction value becomes smaller, that is, when the superscript of ! becomes smaller, the system must wait for these processes to die away by themselves, if the designer doesn't kill them manually.

$$\bar{x}^{*}<!, k>.P1|x^{*}<P>.P2|!^{n}P\rightarrow P1|P2|!^{n+k}P$$

It should be noted that when the superscript of ! is n+K≤0, it means that no P process is allowed to run in the system.

The permission of the number revision in the duplication processes has a high value in the practical use. The strategy can not only properly allocate the uses of the resources

in the system according to the actual situation, but also improve the efficiency and the stability of the system.

For example, $!^{n} P$ is a data transmission process, in which each P is responsible for transmitting one block of the data. Under the premise that the external resources only have to consider about the bandwidth, the increase of the bandwidth available could lead to the corresponding increase of the number in the concurrent P processes through the adjustment of n to transmit data, so that the system efficiency can be achieved. Similarly, under the condition that the cut-off time of the mission can be met, the decrease of the bandwidth available could lead to the corresponding decrease of the number in the concurrent P processes, so that the system stability can be guaranteed.

3.6 The Application of τ to Indicate the Internal Actions or the Intermediate Running State of Related Processes

Although τ is used to signify the internal action in π calculus, in the quantification of the seal calculus, there are some extended applications in the use of τ. A typical example is the local write, in which the syntax of the seal calculus is: $x^{*}(\vec{u}).P \mid \overline{x}^{*}(\vec{v}).Q \to P[\vec{v}/\vec{u}] \mid Q$

The addition of τ turns it into a step-by-step deduction based on the time series by writing into:

$x^{*}(\vec{u}).P \mid \overline{x}^{*}(\vec{v}).Q \to \tau(x^{*}(\vec{u})).P \mid \tau(\overline{x}^{*}(\vec{v})).Q$

$\tau(x^{*}(\vec{u})).P \mid \tau(\overline{x}^{*}(\vec{v})).Q \to P[\vec{v}/\vec{u}] \mid Q$

or writing into:

$x^{*}(\vec{u}).P \mid \overline{x}^{*}(\vec{v}).Q \to <\tau(x^{*}(\vec{u})), X_{\tau 1}>.P \mid <\tau(\overline{x}^{*}(\vec{v})), X_{\tau 2}>.Q$

$<\tau(x^{*}(\vec{u})), X_{\tau 1}>.P \mid <\tau(\overline{x}^{*}(\vec{v})), X_{\tau 2}>.Q \to P[\vec{v}/\vec{u}] \mid Q$

In some specific situations, for instance, when both processes of the communication have no influence on other processes, the intermediate state can be simply written as τ or <τ， X_{τ}>, which is also an internal movement. The following example is such a case.

$P \mid (x^{*} \mid \overline{x}^{*}).Q \to P \mid (\tau(x^{*}) \mid \tau(\overline{x}^{*})).Q$, while $P \mid (\tau(x^{*}) \mid \tau(\overline{x}^{*})).Q \Leftrightarrow P \mid \tau.Q$

In the quantification of the seal calculus, if τ contains channel parameters, in which the channels are used to send and receive actions in pairs, it indicates both processes of the communication are running. If τ appears alone with a single channel parameter, this can indicate that it is an internal action which uses that single channel. If τ does not contain any parameter, this can indicate that it is an internal action which is being carried out, or based on the implication in π calculus, it can be an internal action which hasn't been carried out yet and is on the point of doing so. Through this way, the addition of the operator τ is meant to facilitate the control of the processes which are being carried out in the system in the quantification of the seal calculus.

The following example can well illustrate how the system reasonably makes real-time scheduling or dispatch to the processes under the condition of a dynamic change of a seal resource.

It is assumed that there is a subsystem for a video conference in the information system of the spatial network. In the subsystem, according to the actual situation, two options in video quality can be considered in the use of video transmission: One is cif (512kbps), and the other is d1(1.5mbps). In the normal situation, d1 is usually the quality preferred for video transmission. But there are some particular situations which may lead to the reduction of the bandwidth, like signal attenuation caused by heavy rain in the route of the air-to-ground transmission. Obviously, the reduction of the bandwidth will have some negative effects on the video conference. In order to guarantee the normal running of the

video conference, lowering the video quality through the system scheduling (change the interrupted video transmission from d1 to cif) can keep the conference going on.

For the writing convenience, simplified expressions are adopted in the following example. In other words, a matrix for the resource demands of the process and the resource vectors of the seal are omitted in the system description. Only when much emphasis is placed on the process being incapable of running in the hibernate seal, could there be the necessity to explain that the providing resource is 0.

The communication module of the subsystem for the video conference is illustrated as: $!^{1}\left(s1.c^{*}+s2.d^{*}\right)|!^{1}(\overline{c}^{*}+\overline{d}^{*})$

It is supposed that c stands for the video module with the sending format being cif, whose resource demands for the network is 512 kbps, while d stands for the video module with the sending or receiving format being d1, whose resource demands for the network is 1.5mbps.

Suppose the network bandwidth of the system resources is 10mbps, then the control process of the system transmits the information of s2 to the communication module:

$$!^{1}\left(s1.c^{*}+s2.d^{*}\right)|!^{1}(\overline{c}^{*}+\overline{d}^{*})|\overline{s2}^{*}\text{ .That is:}$$

$$!^{1}\left(s1.c^{*}+s2.d^{*}\right)|!^{1}(\overline{c}^{*}+\overline{d}^{*})|\left(s1.c^{*}+s2.d^{*}\right)|(\overline{c}^{*}+\overline{d}^{*})|\overline{s2}^{*}\text{ .}$$

In this way, the system becomes: $!^{1}\left(s1.c^{*}+s2.d^{*}\right)|!^{1}(\overline{c}^{*}+\overline{d}^{*})|d^{*}|(\overline{c}^{*}+\overline{d}^{*})$

Then the system starts to be into the video conference mode:

$$!^{1}\left(s1.c^{*}+s2.d^{*}\right)|!^{1}(\overline{c}^{*}+\overline{d}^{*})|\tau(d^{*})|\tau(\overline{d}^{*})$$

However, before the end of the video conference, it happens that the heavy rain causes the network to lower the bandwidth from 10mbps to 1mbps. In order to keep the conference going on, the control system now transfers the format from d1 to cif so that the conference will not be interrupted. In order to achieve this goal, the first thing to do is to terminate the video transmission in the d1 format. So the control system must first terminate $\tau(d^{*})|\tau(\overline{d}^{*})$. There are two steps for the termination of the process. The first step is to move the process which is about to be terminated to the seal with the resource being 0. Then the second step is to destroy that seal. After that, the control process sends out to the system the message of Hibernate .

$$x^{*}\{<Hibernate,0>\}.y^{*}\{<Hibernate,0>\}|\overline{x}^{*}(\tau(d^{*})).\overline{y}^{*}(\tau(\overline{d}^{*}))$$

In this case, the whole system becomes:

$$!^{1}\left(s1.c^{*}+s2.d^{*}\right)|!^{1}(\overline{c}^{*}+\overline{d}^{*})|\tau(d^{*})|\tau(\overline{d}^{*})|x^{*}\{<Hibernate,0>\}.y^{*}\{<Hibernate,0>\}|\overline{x}^{*}(\tau(d^{*})).\overline{y}^{*}(\tau(\overline{d}^{*}))$$

Through reaction, the system becomes:

$$!^{1}\left(s1.c^{*}+s2.d^{*}\right)|!^{1}(\overline{c}^{*}+\overline{d}^{*})|<Hibernate,0>[\tau(d^{*})|\tau(\overline{d}^{*})]$$

After freezing the running process and then deleting it, the control process of the system sends out $\overline{k}^{*}\{Hibernate\}|k^{*}\{\}$.

Through reaction, the system becomes: $!^{1}\left(s1.c^{*}+s2.d^{*}\right)|!^{1}(\overline{c}^{*}+\overline{d}^{*})$.

After the deletion of the process, the control process of the system starts up another process c which occupies less network resource. Then, the control process of the system sends s1, and the system becomes $!^{1}\left(s1.c^{*}+s2.d^{*}\right)|!^{1}(\overline{c}^{*}+\overline{d}^{*})|\tau(c^{*})|\tau(\overline{c}^{*})$. After a reasonable dispatch of the system, the conference can go on without any interruption.

4. Conclusion

The quantification of the seal calculus discussed in the paper is an extension of quantified description based on the traditional seal calculus. It is a very successful effort to combine the qualitative and quantitative description together. Many examples illustrated in this paper also prove that the quantification of the seal calculus has enough ability to describe something which cannot be done by the traditional seal. Beside, the quantification of the seal calculus can identify clearly some of the problematic analysis which is correct qualitatively in the traditional seal calculus, but incorrect in deduction after the quantitative analysis. The problematic analysis has become a potential flaw that greatly curbs the application and development of the traditional seal calculus and other formalized modeling languages in actual use.

Therefore, the combination of the qualitative and quantitative description is a necessity in all the formalized descriptive languages including the seal calculus. The reasoning is quite similar to the slow development of algebra and geometry before their combination, because, after the combination, both sciences obtained greater strengths in descriptive and analytic abilities, really encouraging the development of both. Similarly, it is creatively significant to extend the use of the traditional seal calculus to that of the quantification of the seal calculus under the combination of the qualitative and quantitative analysis, which is believed to have far-reaching influences on the development and application of the formalized research.

5. Acknowledgements

The work has been funded by the Program of Critical Theories and Technological Researches on the New Information Network (SKLSE-2015-A-06) of the State Key Lab of Software Engineering, Computer School of Wuhan Univ., China.

References

[1] G. Castagna, J. Vitek, F. Zappa Nardelli. The seal calculus [J]. Information and Computation, 201 (2005).
[2] G.Castagna and J.Vitek. Confinement and commitment for the seal calculus. http://cui.unige.ch/OSG/publications/OO-articles/TechnicalReports/99/commitment.pdf,NOV.1998..
[3] J.Vitek and G.Castagna. seal: A framework for secure mobile computation.[DB/OL] In Internet Programming Languages, number 1686 in Lecture Notes in Computer Science.Springer,1999
[4] J.Vitek and G.Castagna. A calculus of secure mobile computations In Proceedings of IEEE Workshop on Internet Programming Languages,(WIPL).Chicago,I11.1998.
[5] Francesco Zappa Naderll. Types for seal calculus. Forthcoming master Thesis, Octomber 2000
[6] L. Cardelli and A.D. Gordon. Mobile ambients. In M. Nivat, editor, FoSSaCS 1998, volume 1378 of Lecture Notes in Computer Science, pages 140–155, 1998.
[7] L. Cardelli and A.D. Gordon. Types for mobile ambients. In POPL 1999, pages 79–92. ACM,1999.
[8] L.Cardelli and A.D. Gordon. Anytime, anywhere: Modal logics for mobile ambients. In POPL2000, pages 365–377. ACM, 2000.
[9] L.Cardelli and A.D.Gordon. Mobile ambients.Theoretical Computer Science, 240(1):177–213,2000.
[10] L.Cardelli. Mobile ambients. In Secure Internet Programming: Security Issues for Mobile andDistributed Objects, volume 1603 of LNCS, pages 51–94, 1999.
[11] R. Milner. Communicating and Mobile Systems: the π-calculus[M]. Cambridge: Cambridge University Press, 1999.
[12] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, Parts I and II. Information and Computation, 100:1–77, September 1992.
[13] R. Milner. The polyadic-calculus: a tutorial. Logic and Algebra of Specification,94:91–180, 1993.R. Milner. The Polyadic π-Calculus: a Tutorial. Theoretical Computer Science,1988.

Author

Nan Zhu, assisted professor, Research direction: computer software.