The Quantification of Seal Calculus in the English Simulation Software System

Qiaohong Wang

Zhengzhou University of Industrial Technology, Henan, China.

Abstract - **Seal calculus is a tool targeting the modeling of mobile systems. 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 analysis concerns not only the internal business logic of the system, 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 closely the actual model as possible, this paper introduces the concept of quantification analysis to be carried out 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 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.

I. INTRODUCTION

As described by Castagna et al [1] the Seal Calculus is a process language for describing mobile computation. Threads and resources are tree structured; the nodes thereof correspond to agents, the units of mobility. The Calculus extends a π-calculus core with synchronous, objective mobility of agents over channels.

In the design and development of English software system, it is crucial to establish a scientific English software model. Since the English 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 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 English 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.

II. A BRIEF DESCRIPTION OF SEAL CALCULUS

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

Assuming η_1 and η_2 are two locations, and y is a name, then we can define:

Shared channels:

$$\text{Synch}_y^S(\eta_1,\eta_2) = (\eta_1 = y \wedge \eta_2 = \uparrow)$$

Located channels:

$$\text{Synch}_y^L(\eta_1,\eta_2) = (\eta_1 = y \wedge \eta_2 = *) \vee (\eta_1 = * \wedge \eta_2 = \uparrow)$$

Deductive rules for the seal calculus:

Local writing:

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

Write in:

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

$$P \mid y[(v\vec{z})(Q_1[\vec{w}/\vec{u}] \mid Q_2)]$$

Write out:

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

$$(v\vec{v} \cap \vec{z})(P[\vec{v}/\vec{u}] \mid y[(v\vec{z} \setminus \vec{v})(Q_1 \mid Q_2)])$$

Local movement:

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

Move in:

$$\overline{x}^{\eta 1}\{v\}.P \mid v[S] \mid y[(v\vec{z})(x^{\eta 2}\{\vec{u}\}.Q_1 \mid Q_2)] \rightarrow$$

$$P \mid y[(v\vec{z})(Q_1 \mid Q_2 \mid u_1[S] \mid \ldots \mid u_n[S])]$$

Move out:

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

$$(vfn(R) \cap \vec{z})(u_1[R] \mid \ldots \mid u_n[R] \mid y[(v\vec{z} \setminus fn(R))(Q_1 \mid Q_2)])$$

III. 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.

A. 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.

B. 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:

$$P3 \mid y[\overline{a}^{*}.P1 \mid a^{*}.P2] \rightarrow P3 \mid y[P1 \mid P2],$$

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,

$$P3 \mid < y, F_y > [< \overline{a}^{*}, X_{a1} > .P1 \mid < a^{*}, X_{a2} > .P2] \not\rightarrow P3 \mid < y, F_y > [P1 \mid P2]$$

Under the condition when ambiguity is not likely to happen, in order to write conveniently, the expression above could be written as:

$$P3 \mid y[\overline{a}^{*}.P1 \mid a^{*}.P2] \not\rightarrow P3 \mid y[P1 \mid P2].$$

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:

$$!apply(m\ c).(open_c\ \overline{m} \mid m) \quad (2)$$

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, $R_0 \ldots R_{m-1}$ 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

$$\begin{matrix} r_{00} & \cdots & r_{0\,m-1} \\ \vdots & \ddots & \vdots \\ r_{n0} & \cdots & r_{n\,m-1} \end{matrix}$$

Figure 1

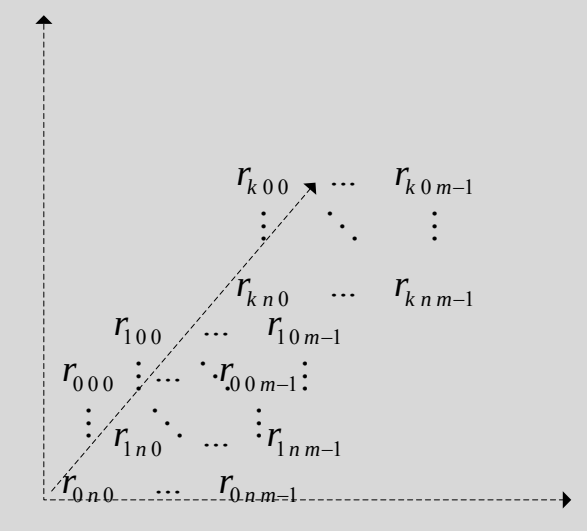

Figure 2

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.

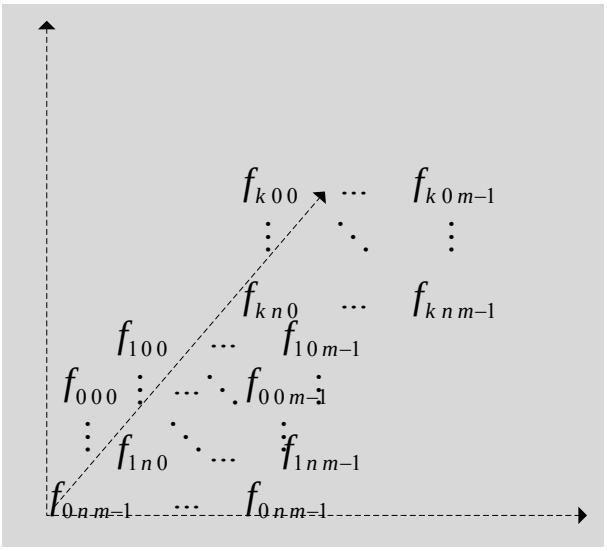

Figure 3

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.

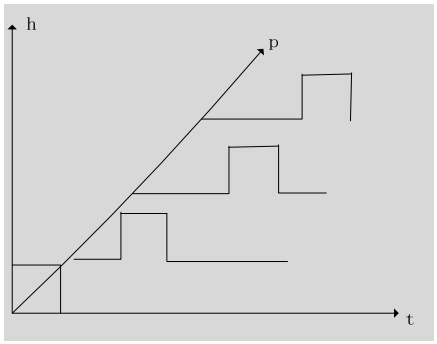

Figure 4

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.

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).

IV. CONCLUSIONS

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.

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 English 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, October 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