A Simple Modeling and Realization Based on π-Based Comparison Algorithm

Hao Bu[123]*, Shihong Chen[13] , Rong zhu[12], Xiaoqiong Tan[23]

[1]Computer School of Wuhan University, China

[2]Collaborative Innovation Center of Geospatial Technology, China

[3]National Engineering Research Center for Multimedia Software, Computer School of Wuhan University, China

plutohb@hotmail.com

Keywords: π calculus; well-orderedness; movement calculation; comparison

Abstract. As a very efficient tool, π calculus could do modeling and make evaluations effectively particularly by targeting the features of concurrency and mobility which quite commonly exist in the software system nowadays. After a brief introduction of π comparison, this paper mainly discusses a comparison algorithm based on π comparison which is frequently used in the software modeling. As the ways of data storage mainly concern about random storage of elements and linked list storage of elements, this paper analyzes the algorithm of comparing any two elements in both ways. Particularly, the efficiency analysis is carried on to the element comparing algorithm in the linked list storage, which is relatively complicated, finding that the designed algorithm in this paper is greatly improved not only in algorithm simplification but also in working efficiency.

1. Introduction

It is generally acknowledged that an unambiguous formal description is the first procedure to start solving any problem by using the computer. But the features of concurrency and mobility in modern software have made the traditional functional theory obsolete from dealing with the description of this kind of system. Under this situation, new methods are urgently needed and hence appeared π calculus, which has been promoted by Prof. Robin Miller (Turing Prize winner) and his collaborator. On the base of the Calculus of Communicating System (CCS), π calculus, a new concurrency theory, is formed by focusing on the moving communication between processes as the research subject and by supporting the modeling and synchronic testing of the dynamic system. Π calculus has the advantage of dealing with all the things in the process by names without distinction, no matter whether they are data values, data variables, input parameters or the channels used for transmitting data. This kind of expression by using names as the basic conceptions makes it possible for a complicated mobile system to be presented only in some simple mathematical structures.

The element comparison is one of the most basic calculations which one may frequently encounter in all kinds of systems, but unfortunately π calculus itself doesn't offer the comparing operators. So how to deal with the modeling problems based on π calculus in actual systems is the discussion focus of this paper, which attempts to look for the solution through the discussion of the comparison calculation of any two elements in a well-ordered set.

2. A brief introduction of π-calculus

2.1. Bisimulation

As the theoretical foundation of π-calculus, bisimulation actually is a kind of behavior equivalence of a process. If the behaviors of two processes are identical, then one process could be replaced by the other process in any system. As the key for π-calculus to be widely applied, the bisimulation theory is the basis of the equivalence theory of π-calculus. Its definition is presented as follows:

Bisimulation is a kind of symmetric binary relation. That is, if $P \Re Q$ and $P \xrightarrow{a} P'$, then there is a Q' to satisfy $P' \Re Q'$ and $Q \xrightarrow{a} Q'$.

Bisimulation can be classified into strong ones and weak ones. If there is a strong relation of bisimulation in two processes, then the two processes can be called strong equivalence. If there is a weak one in two processes, then they are called weak equivalence. Strong bisimulation between the processes requires the consistency between the internal and external actions, while the two processes with weak bisimulation may be just identical with the external actions, but not with the internal communication action. In practical application, what we concern more is whether there externally exist the same behavioral capabilities in the two processes, which means that weak bisimulation is more widely applied. The relation between deduction and reduction of the system mentioned in the following part is based on weak bisimulation, while that between the pre-deductive system and the post-deductive system is also based on weak bisimulation.

2.2. Interaction

In π-calculus, processes represent the units for concurrent operation entities, and each process has a number of channels for communication with other processes. Since both data and the channels used for delivering data are unanimously defined by names without any differentiation, processes and names are the two basic entities in π-calculus. So the interaction of π-calculus actually means data for two processes or two systems are communicated by using channels.

π-Calculus can be classified into monadic π-calculus and polyadic π-calculus. The former allows the number for an input or an output to be just one, while the latter allows the number for it to be either zero or multiple. Up to now, in comparison with the original definition of π-calculus, though there are many extensions, such as high order π-calculus and asynchronousπ-calculus, the essence of π-calculus hasn't gone through substantial changes.

The simple version of π-calculus interaction is illustrated in Figure 1.

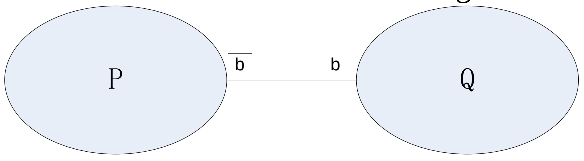

Figure 1. The simple version of π-calculus interaction

In Figure 1, the process P goes through the port b down the channel to send a message to the process Q, which receives it through the channel.

The concurrent version of π-calculus interaction is illustrated in Figure 2.

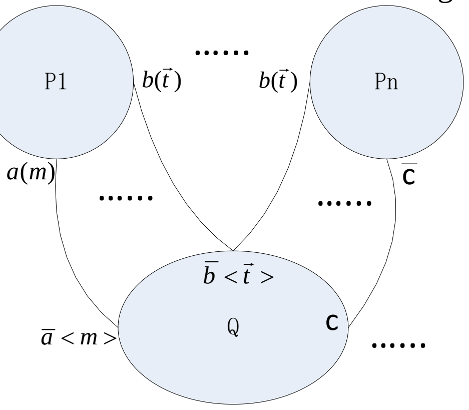

Figure 2. The concurrent version of π-calculus interaction

In Figure 2, the process Q goes through the port a to send the message m to P1, which receives it through the same port from the same process Q. The process Q sends the message sequence $\vec{t}$ to the process P1 and the process P2 through the port b, so that both processes can receive the message sequence $\vec{t}$ from the process Q through the port b. The process Pn sends an empty message to the process Q through the port c. The punctuation ellipsis "……" means the omitted processes or channels.

2.3. π-calculus syntax

In π-calculus, sending or receiving a message (or a name) or making a silent transition could be represented by action prefixes. Its syntax and corresponding meanings are[1,2,3]:

π ::= m(n) receive the limited name n from the channel m.

m($\vec{n}$) receive the limited name sequence $\vec{n}$ from the channel m (Notes: the number of the receiver and the sender must be the same) .

$\overline{m}$<n> send the free name n from the channel m.

$\overline{m}<\vec{n}>$ send the free name sequence $\vec{n}$ from the channel m (Notes: the number of the sender and the receiver must be the same).

τ the internal action within the process, invisible from the outside.

The process P in π-calculus can be syntactically defined as:

$P ::= \sum_{i\in I} \pi_i.P_i \,|P_1\,|\,P_2\,|\,new\ n\ P\,|\,!P$

$\sum_{i\in I} \pi_i.P_i$ is termed as a sum of the processes, in which I is the finite subscript set.

In the sum of the processes, π is guarded by π_i. That is, π has to start its action right after the action represented by π_i is finished.

The p1|p2 means the concurrent operation of the process P_1 and the process P_2.

The new n P means that the name n included in the process P is highly restricted or limited. Different from any other name outside P, this new n is quite unique even if it is still called n. For example,

$P = new\ a(a.Q + b.R)\,|\,\overline{a}.0)\,|\,(\overline{b}.S + \overline{a}.T)$ could lead to the two following deductions.

$P \rightarrow new\ a(Q)\,|\,(\overline{b}.S + \overline{a}.T)$

$P \rightarrow new\ a(R)\,|\,\overline{a}.0\,|\,(S)$

But the following deduction is wrong.

$P \not\rightarrow new\ a(Q)\,|\,\overline{a}.0)\,|\,(T)$

!P means the concurrent operation of any multiple duplicates of P.

Similar to the sum of processes, $\prod_{i\in I} P_i$ could be used to mean the concurrent execution of multiple processes in order to achieve the conciseness of expressions, so the syntactic expression could be presented as:

$P ::= \sum_{i\in I} \pi_i.P_i \,|\prod_{i\in I} P_i\,|\,new\ n\ P\,|\,!P$

3. The comparison of any two elements in the well-ordered set by using random storage of π calculus

In a well-ordered set, when two elements are selected for sequential comparison, if one wants to get the feedback information of the sequence for these two elements, different algorithms will be adopted according to different ways of data storage. In the randomly selected way of data storage, every element saves the sequential information with all the other elements. We can suppose that there are elements [a1..an] in a set. Meanwhile, without loss of generality, it is supposed that the sequential relations between elements happen to be consistent with the sequence of element subscripts. In other words, if k<h, then ak<ah. So if am is taken as the example, then we can express its storage as follows:

$$A_m = \sum_{i=1}^{k-1} a_i.greater + a_m.equal + \sum_{j=k+1}^{n} a_j.less$$

If the comparison is between am and an, then it can be described as $A_m\,|\,\overline{a_n}$. If the result is "greater", it means that am is before an. If the result is "less", then it means that am is after an.

The advantage of this way is that the storage has quicker speed with relatively less use of computation resources, but its disadvantage is that the storage is much more complicated. After all, the storage efficiency is achieved at the expense of complexity, which will be illustrated not only in storage, but also in the data maintenance. Therefore, when the data need some adjustments like addition, deletion, changes or checkups, the maintenance requires more efforts to be carried on. But this kind of expression has its own merit: the expressing flexibility, which can illustrate not only the hierarchical structures between data, but also the complex network structures between data.

4.The comparison between any two elements in the well-ordered set stored in the manner of a linked list under the realization of π calculus

When the well-ordered set is saved in the structure of a linked list, the comparison algorithm between two elements will be relatively more complicated.

The illustration of the linked list is as follows in Figure 3:

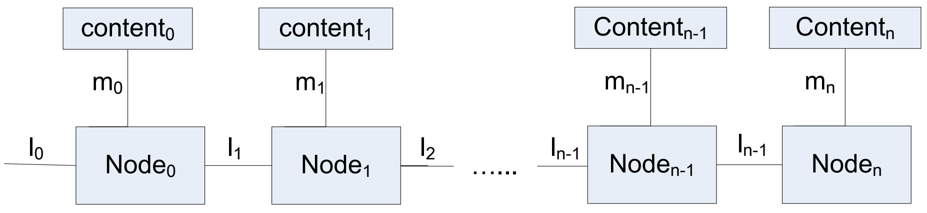

Figure 3. linked list

The method mentioned in "The Comparison Realization of the Expressions Based on π-calculus" goes like this: the comparison starts first from looking for A; the successful location of A will lead to the following search for B according to the sequence. If B is found, that means the position of B is after that of A. In another word, B is larger than A. However, if nothing could be found till the end of the sequence, which means the position of B is before A. That is, B is smaller than A. [4]

However, the method used in this paper is different from the above-mentioned one. The main idea is to let two elements start simultaneously from the beginning to the end of the linked list by following the sequence, so that a match could be found. Once some element matches successfully, it will give the feedback information of the compared result.

In comparison, five processes are needed.

The first is the well-ordered process"Welord", inside which the well-ordered set is saved. The stored set sequence is taken as the standard for the comparison of the two elements.

The second is the initiating process "Launch" which saves the position information of both the initiating signal and the first node of the well-ordered set. So its function is to initiate the comparison of the elements.

The third is the postman process "Postboy", whose function is to take out elements in succession from the well-ordered process to make a match with elements in the Analyse process.

The fourth is the analyzing process "Analyse", whose function is to match and analyze the information of the elements received from the Postboy process.

The fifth is the result process "Result" to show the compared result.

The well-ordered process: Welord = $\prod_{i=0}^{n}(!l_i(c_{i+1}).\overline{c_{i+1}}(a_i l_{i+1}))$

The main names and their related functions:

$l_i(c_{i+1})$: l_i is the node pointer. When i=0, l_0 is regarded as the initial pointer to use l_i channel to receive C_{i+1} from the Postboy process.

$\overline{c_{i+1}}(a_i l_{i+1})$: it will use the received C_{i+1} as the channel so as to send a_i l_{i+1} to the Postboy process. In the expression, a_i is the compared data,while l_{i+1} is the pointer towards the end of the linked list.

The initial process: Launch = $\overline{switch}.\overline{r}(l_0)$

The main names and their related functions:

switch is the on/off switch used to start the matching operation of the Postboy process.

r is used to deliver the node information of the Welord process to the Postboy process. In the expression, $\overline{r}\left(l_0\right)$ is to send the initial node pointer of the Welord process l_0 to the Postboy process.

The postman process: Postboy= $!switch.r(m).\overline{m}(pc\ ).(pc\ (f\ e).f.\overline{r}\ (e))$

Each time the Postboy process takes out an element from the Welord process, its successful match with the Analyse process will lead to the return of an on/off switch value to initiate the match for the next round.

The main names and their related functions:

switch is the on/off switch used to protect the Postboy process for unlimited self-matching. Only when the process Analyse sends a message of releasing the guard can the Postboy process continue the step of self-matching.

r is used to send the information of the Welord node to the Postboy process itself.

f is the element content and e is the information of the next node pointer.

The analyzing process: Analyse= $\prod_{k=0,k\neq j,k\neq s}^{n} \overline{a_k}.\overline{switch} \mid (\overline{a_j}.\overline{result\_j} + \overline{a_s}.\overline{result\_s})$

The main names and their related functions:

a_k is used to make the matching operation with the corresponding element in the Welord process. The completion of the matching operation will be followed with the switch information being sent to release the guard of the Postboy process once.

switch is a on/off switch used to initiate the matching operation of the Postboy process.

r is used to send the node information of the Welord process to the Postboy process. The information includes $\overline{r}\left(l_0\right)$ the initial node pointer of the Welord process l_0 to the Postboy process.

The result display process: Result= $result\_j \mid result\_s$

Suppose, in the result display process, the value result_j indicates that a_j is behind a_s, then the value result_s indicates that a_s is behind a_j.

The description of calculation for the whole comparison is: welord|Launch| Postboy |Analyse|Result

Its specification process is as follows:

1) First initiate the process launch to open the switch of the postboy process P. Then the initiating process launch becomes: $\overline{r}\left(l_0\right)$

The Postboy process P becomes:

$r(m).\overline{m}(pc\ ).pc\ (f\ e).f.\overline{r}\ (e) \mid !switch.r(m).\overline{m}(pc\ ).pc\ (f\ e).f.\overline{r}\ (e)$

2) Then start the initiating process launch to send the initial pointer l_0 to the Postboy process P to complete the initiating process.

The Postboy process P becomes:

$\overline{l_0}(pc\ ).pc\ (f\ e).f.\overline{r}\ (e) \mid !switch.r(m).\overline{m}(pc\ ).pc\ (f\ e).f.\overline{r}\ (e)$

3) The Postboy process P depends upon l_0 channel to send pc to the Welord process. The Postboy process P becomes:

$pc\ (f\ e).f.\overline{r}\ (e) \mid !switch.r(m).\overline{m}(pc\ ).pc\ (f\ e).f.\overline{r}\ (e)$

The Welord process becomes: $\overline{pc}(a_0 l_1) \mid \prod_{i=0}^{n}(!l_i\,(c_{i+1}).\overline{c_{i+1}}(a_i l_{i+1}))$

4) The Welord process uses the pc channel to send (a_0 l_1) to the Postboy process. Moreover, it changes the name of (f e) of the Postboy process into (a_0 l_1 respectively. Now the Postboy process becomes:

$a_0.\overline{r}\ (l_1) \mid !switch.r(m).\overline{m}(pc\ ).pc\ (f\ e).f.\overline{r}\ (e)$

But this time the Analyse process sends a_0 to the Postboy process. Suppose both j and s are not equal to 0, then the Analyse process becomes:

$$\overline{switch} \mid \prod_{k=1,k\neq j,k\neq s}^{n} \overline{a_k}.\overline{switch} \mid (\overline{a_j}.\overline{result\_j} + \overline{a_s}.\overline{result\_s})$$

The Welord process becomes: $\overline{r}(l_1) \,|\, !switch.r(m).\overline{m}(pc\ ).pc\ (f\,e).f.\overline{r}(e)$

5) Then, the Analyse process sends the switch information to the Welord process. The Analyse process becomes:

$$\prod_{k=1,k\neq j,k\neq s}^{n} \overline{a_k}.\overline{switch} \,|\, (\overline{a_j}.\overline{result\_j} + \overline{a_s}.\overline{result\_s})$$

The Welord process becomes: $\overline{r}(l_1) \,|\, r(m).\overline{m}(pc\ ).pc\ (f\,e).f.\overline{r}(e) \,|\, !switch.r(m).\overline{m}(pc\ ).pc\ (f\,e).f.\overline{r}(e)$

6) Through internal stipulations, then the Welord process becomes:

$\overline{l_1}(pc\ ).pc\ (f\,e).f.\overline{r}(e) \,|\, !switch.r(m).\overline{m}(pc\ ).pc\ (f\,e).f.\overline{r}(e)$

This is equal to the movement from the node l_0 tol_1.

7) Following the deduction like this, without loss of generality, suppose the match has come to a_j, then the Postboy process becomes:

$\overline{r}(l_j) \,|\, !switch.r(m).\overline{m}(pc\ ).pc\ (f\,e).f.\overline{r}(e)$

Then the Analyse process has become: $\prod_{k=j-1,k\neq j,k\neq s}^{n} \overline{a_k}.\overline{switch} \,|\, .\overline{result\_j}$

Since the Analyse process doesn't send the switch information to the Welord process, it leads to stopping the match between the Postboy process and the Welord process.

8) The Analyse process sends result_j to the result display process, then the result display process becomes $result\_s$, which indicates the position of a_s is behind a_j.

5. Conclusion

The comparison between elements is a technique frequently used in developing concurrent system modeling. Although π calculus doesn't provide comparing operators, the paper makes up for the weakness by offering the algorithm for corresponding element comparison on the base of different manners of element storage. The algorithm is relatively easier for the element comparison in the manner of random storage, but it is much more complicated for the element comparison in the set stored in alignment or linked list. Taking advantage of the special feature of π calculus, this paper greatly improved the efficiency of the algorithm.

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 University, China.

References

[1] Milner, Bobin. “Communicating and Mobile Systems: the π-calculus”. Cambridge, UK: Cambridge University Press, **(1999).**

[2] R. Milner. “The polyadic-calculus: a tutorial”. Logic and Algebra of Specification,94:91–180, **(1993).**

[3] R. Milner. “The Polyadic π-Calculus: a Tutorial”. Theoretical Computer Science,198:239–249, **(1997).**

[4] H. Bu, S. Chen “The comparison realization of the expressions based on π-calculus” 33:1281-1296,**(2015)**.

[5] X. Liu and D. Walker. “A Polymorphic Type System for the Polyadic pi-calculus”.In Proceedings of the 6th International Conference on Concurrency Theory, pages 103–116. Springer-Verlag London, UK, **(1995)**.

[6] Jing Hongye. “Veritfication on the Web Service Composition Based on Pi-Calculus” Graduation thesis Taiyuan University of Technology**(2008)**.

[7] Cai Xiaojuan. “The Expressiveness of π-Calculus via Programming ” Dissertation Shanghai Jiao Tong University**(2009)**.

[8] Hao Kegang. Guo Xiaoqun Li Xianglin “The Pi+ Calculus - An Extension of the Pi Calculus for Expressing Petri Nets ”. Chinese Journal of Computers, (**2011)**. 34(2): 193-203.

[9] Kang Hui, Zeng Yingyingm, Liu Zhiyong. “Modeling the Mobile Communication Service Based on PI-calculus”. Journal on Communications, (**2009)**, 30(4):11-16.

[10]Wei Yandong. “Research on formal verification of web services flow based on pi-calculus” Graduation thesis Zhejiang University**(2008)**.

[11]Yu Pengwei. “WS-BPEL Modeling and Realization Based on π-Calculus” Graduation thesis Northwest University(**2009)**.

[12]J. Liu, H. Lin, “A complete symbolic bisimulation for full applied pi calculus”, Theoret. Comput. Sci. 458 **(2012)** 76–112.

[13]J. Dreier, C. Ene, P. Lafourcade, Y. Lakhnech, On unique decomposition of processes in the applied pi-calculus, in: Proceedings of the 16th International Conference Foundations of Software Science and Computation Structures, FOSSACS’13, Rome, Italy, in: LNCS, vol. 7794, Springer, (**2013)**, pp. 50–64.

[14]R. Demangeon, K. Honda, Full abstraction in a subtyped pi-calculus with linear types, in: CONCUR, in: LNCS, vol. 6901, Springer-Verlag, (**2011)**, pp. 280–296.

[15]DimitrisMostrousa, NobukoYoshidab, ession typing and asynchronous subtyping forthehigher-order π-calculus, Information and Computation 241 **(2015)** 227–263