并行计算模型对比分析*)

王 欢 都志辉
(清华大学计算机系 北京 100084)

摘 要 随着集群式系统的发展,并行计算模型在估计和评价系统的性能、引导集群的体系结构以及指导并行算法和程序的设计等方面都显得越来越重要。对于目前已有的并行计算模型的设计思想和原理的了解和分析,非常有利于新的模型的设计与研究。本文首先介绍了目前比较常见的5种并行计算模型,接着在同步性、通信方式和参数等3个方面分析比较了它们的异同和优缺点,最后得出结论,指出了下一代并行计算模型的发展趋势是与具体应用相关的并行计算模型。

关键词 PRAM,BSP,Postal Model,LogP,QSM,并行计算模型

Contrastive Analysis of Parallel Computation Model

WANG Huan DU Zhi-Hui
(Department of Computer Science and Technology, Tsinghua University, Beijing 100084)

Abstract With the rapid development of the clusters, parallel computation model is more and more importance on the design of the parallel algorithms and the parallel program. It is quite beneficial for the further study of new parallel model to understand and analyze the designing concept and rationale of the existing ones. This paper introduces five parallel computation models which are well known to the people firstly. Then, the difference of the style of synchronization and communication is compared in this paper. Finally, it is concluded that the next generation of parallel computation model may be relative with the actual computation pattern.

Keywords PRAM, BSP, Postal Model, LogP, QSM, Parallel computation model

1 引言

随着集群式计算机系统NOW[1]的发展,集群式计算在超级计算机研究领域产生了巨大的影响,集群式系统将是未来超级计算机的主流结构形式。推动集群式系统发展的两大关键技术是网络和CPU,相对而言网络的作用更大,而集群的整体性能提高正是在新型的网络互连技术的推动下而获得的。这种通过高速网络互连的体系结构已经从根本上动摇了原来超级计算机体系结构的格局,在一定程度上可以说内部互连网络是超级计算机的枢纽和核心。超级计算机发展到现在,从体系结构到算法,到程序设计已经发生了巨大变化,这一切和内部互连网络结构的发展密切相关。

然而,目前的集群式系统中,还没有一个非常合适的能够刻画内部互联网络特点的通信模型。本文旨在对目前已经有的模型做一个分析比较,希望在此基础上能提出更好的模型来刻画集群式系统的性能,同时对系统和软件的设计带来更多的指导。

2 现有并行计算模型简介

并行计算机的发展是共享存储到分布式存储、从同步到异步。与此同时,并行计算模型的发展也经历了这个过程,下面介绍一下已有的各个模型。

2.1 PRAM(Parallel Random Access Machine)及其扩展模型

PRAM是一种理想的并行计算模型。一台PRAM并行计算机由若干带有局部存储器的处理机和一个全局的共享存储器构成。

一个PRAM模型的计算由一连串的读、计算和写序列组成。在读步骤中,每个处理器从全局内存中读取数据到局部内存中。在计算步骤中,每个处理器处理各自局部内存中的数据,并把结果存放在局部内存中。在写步骤中,每个处理器将各自局部内存中的结果写入相应的全局内存中。为了解决处理机时间读、写冲突,又可按照处理机对共享单元存、取的不同约束条件进一步分为如下几种情况:

(1) EREW PRAM (Exclusive Read Exclusive Write PRAM)模型。每次只允许一台处理机读或写某一共享单元;

(2) CREW PRAM (Concurrent Read Exclusive Write PRAM)模型。每次允许多台处理机同时读同一个共享单元内容,但每次只允许一台处理机向某个共享单元写内容;

(3) CRCW PRAM (Concurrent Read Concurrent Write PRAM)模型。每次允许多台处理机同时读写同一共享单元内容。

另外在参考文[5]中,又提出了一种新的模型QRQW PRAM(Queue Read Queue Write PRAM),每一个共享单元可以同时被多个处理机读写,这些读写操作被放在一个队列中依次进行。这样,读写一个共享单元的最坏时间就与同时读写该地址的处理机数呈线性相关,一定程度上把同步开销抽象到了模型中。

2.2 BSP(Bulk Synchronous Parallel)模型

根据BSP[4]模型,一个并行计算机由下面3部分组成:第一,若干个存储器或者处理机组件;第二,这些组件之间的点对点通信;第三,这些组件之间的同步机制。

为简单起见,可以认为每个组件中包含一个处理机和本地存储器;在模型中,不要求关于通信系统、互连网络和同步系统的额外信息。

*)本项目受到北京市自然科学基金(项目编号:4042018)、863项目(项目编号:2004AA104330)和973(项目编号:2004CB217903)的资助。王欢 硕士研究生,主要研究方向为并行与分布式计算;都志辉 博士后,副教授,研究方向为并行计算、网格计算。

在 BSP 模型中，一个并行系统由下面 3 个参数来表示：

(1)P，系统中处理机的数目；

(2)g，把通信开销转换为计算开销的因子；

(3)L，两个同步之间的最短时间。

连续的两个同步之间的周期被定义为超步(superstep)。在一个超步中一个处理机可以进行 3 个操作：首先各处理机处理本地存储器中的数据，可以是本地计算；然后各处理机向别的处理机提出远程内存读写请求，而通信实际发生在超步中的时间是不可预知的；最后，所有处理机进行路障同步，本次超步的数据通信仅当同步以后有效。

在 BSP 模型中，计算操作的总量用处理机在计算过程中所做的基本操作的数目来表示，通信量则用字数来表示。假设在同一次计算和通信中数据项的大小是一样的，而且假设在一个超步中的通信总量是被发送或者接受的最大字数。通过因子 g，通信开销被转换成计算开销，其中 g 是系统中计算带宽与通信带宽的比值。具体地说，就是每个处理机发送或者接收至多 h 个字的开销与在相同的时间内进行 gh 次计算操作。L 表示连续的两次同步之间所能执行的操作的次数，它的具体值由同步操作的实现和上层算法来决定。

同步操作使得超步之间相互无关，根据 BSP 模型实现的算法的总开销就是所有超步开销的总和，而每一个超步的开销由该超步中计算开销和通信开销所决定。

2.3 The Postal Model 模型

Postal[6]模型主要用来描述具有下面 3 个方面特点的消息传递的通信系统：完全连通、同时 I/O 和通信延迟。一个带有 n 个处理机和通信延迟 λ 的消息传递系统 MPS(n,λ)有下面 3 个属性：

(1)完全的连通性。系统中的每一个处理机能够向系统中的任何其他处理机发送点对点消息；

(2)同时的 I/O。一个处理机 p 可以在给处理机 q 发送消息的同时接收处理机 r 发送来的消息；

(3) 通信延迟。如果在 t 时刻处理机 p 向处理机 q 发送我消息 M，则 p 在时间间隔$[t,t+1]$内忙于发送消息 M，而且 q 在时间间隔$[t+\lambda-1,t+\lambda]$内忙于接收消息 M。

上面是从处理机的观点来分析一个消息传递系统。在这样的一个系统中，处理机之间靠通过一个通信网络互相发送和接收消息来通信，这样就产生了一个抽象的全连通的系统。而在很多系统中，通过不同的输入输出端口，处理机确实可以同时发送和接收消息。

在 Postal 模型中，message 被用来表示一个在处理机之间进行通信的不可分割的数据单元，一个 message 在发送的时候、传输的时候和接收的时候都不能被分成更小的快。一个原子 message 被定义为一个数据大小的单元，而发送或者接收一个 message 的时间被定义为一个时间单元。发送大块数据的时候，这些数据会首先被分成多个 message，每个 message 都被单独地发送和接收，这在许多报文交换系统(packet-switching)中都是一种标准的实现。

上面所说的通信延迟中包括各方面的系统开销，具体来说包括消息准备时间、输出缓冲拷贝时间、输出端口提交延迟、网络传输延迟、输入端口延迟、输入缓冲拷贝时间和消息中断时间。这里的通信延迟 λ 还包括所有的软件开销和硬件开销。从形式上来说，λ 的值为消息 M 的发送方开始发送消息到接收方完全接收完消息所用的时间。尽管从形式上来说是这样，可实际 λ 的精确值可能取决于实际的发送接收组和实际通信网络的负载。通常，λ 的值应该是相对固定不变的，不同的处理机之间不能有很大的浮动。

2.4 LogP 及其扩展模型

LogP[7]是一个能很好地符合并行计算机系统中这种分布式存储、多处理器网络通信机制的计算模型。利用 LogP 模型，通过 4 个重要参数就可以设计良好的适应不同处理器的算法。这 4 个参数及含义如下：

(1)P(处理器/存储模块的个数)　设处理器进行本地操作的单位时间为一个周期；

(2)L(最大通信延迟)　即从源模块到目的模块传输包含一个字或几个字的信息所用的时间；

(3)o(通信开销)　即一个处理器发送一个信息所用的时间长度，在此时间段中处理器不能执行其它操作；

(4)g(通信间距)　即一个处理器连续进行信息收发的最小时间间隔，g 的倒数对应于通信带宽。

在 LogP 模型中，通信网络被认为有有限的能力，因此在某一个时刻至多只有$\lceil L/g \rceil$消息从任何处理机到任何别的处理机。如果某个处理机尝试传输消息时超过这个限制，这个超出的消息将被延迟，直到网络中别的消息已经传送完毕后再传输。

性能参数 L、o 和 g 都用处理机操作的周期数来表示。LogP 模型是异步的，延迟也只有一个上限，按一定的顺序发出的消息到达时的顺序可能不一致。并且假设所有的消息都是短消息。这些参数的选择是在忠实地获得实际机器的执行特点、提供一个有效的算法设计和分析的框架之间取一个折衷。

参数取得太少，不能很好地描述系统的特点；但参数取得太多，又会使算法的设计变得异常困难。在不同的情况下，LogP 模型中的各个参数也可以适当地化简来降低系统的复杂性。这是因为在不同的情况下，各个参数的重要性也是不一样的，在某些情况下可以忽略一个或者更多参数，使模型变得更简单。比如说，在不常进行通信的算法中，可以忽略带宽和网络能力有限带来的影响；在某些算法中，消息是由长的数据流以流水线的方式通过网络，这可以认为消息的传输时间主要受 g 的限制，而不用考虑通信延迟 L；而在某些机器中，系统的瓶颈是 o，则可以认为 $g=o$，这样就可以不考虑 g 对算法的影响。

LogP 模型通过抽象出来的 4 个参数来评价一个互连通信网络的性能，使得算法设计者不需要关心具体的网络信息，可以更容易地设计符合这种模型的算法。

在 LogP 模型的基础上，又提出了很多扩展模型，主要包括 LogGP[2]，LogGPS[8]，LogPQ[9]，LoGPC[10]。

在 LogP 模型中，并不对长消息做特别的处理，但是现在许多并行计算机对长消息都提供了特别的支持，比如 IBM 的 SP2 等。与发送短消息相比，许多并行机采用处理长消息的方法来达到更高的带宽。因此，在 LogGP 模型中引入了对长消息的支持，它比 LogP 多一个参数 G，代表长消息每字节间距，即发送长消息每字节所需要的时间，它的倒数对应于处理机发送长消息时的带宽。

LogGPS 模型又在 LogGP 模型的基础上加入了同步机制，它定义了一个参数 S，表示消息长度的阈值。当消息长度大于 S 时，处理机之间就要发送和接收同步消息了。

LogPQ 模型则是把通信队列考虑进了 LogP 模型，它在 LogP 模型的基础上加入了 3 个队列：第一，在每个处理机的发送通道上加入了一个发送队列(SQ)；第二，在每个处理机的接收通道上加入了一个接收队列(RQ)；第三，在接收通道的集中点之前有一个全局的传输队列(TQ)。

LoGPC 则是在 LogGP 的基础上加入了通信网络的争论

对性能的影响和网络接口 DMA 行为，这样更实际地分析消息传递系统的性能。

2.5 QSM 模型

QSM(Queue Shared Memory)[11]模型是一个适用于大块同步算法的多指令、多数据流(MIMD)机器的共享存储模型，由若干个处理机组成，各个处理机带有自己的本地存储器，处理机之间通过读写全局存储器来相互通信。每个处理机都执行一个同步步骤队列，每个同步步骤中包含下面 3 个子步骤：

(1)读操作(Read)。每个处理机 i 把全局存储器中内容 r_i 复制到它的局部私有存储器，这里全局存储器中的地址是在每个步骤开始的时候就给出；

(2)计算操作 (Compute)。各个处理机分别进行本地计算 c_i，这些计算只涉及到该处理机的私有存储器和私有状态；

(3)写操作 (Write)。每个处理机 i 把局部私有存储器中的内容写回到全局存储器中 w_i，写入的位置也是在步骤开始的时候就知道了。

对于一个相同的全局地址来说，允许多个处理机对它进行同时的读或者写，但是不允许同时读写。当有多个处理机对同一全局地址进行写操作时，QSM 模型中采用任意选择一台作为优胜者来写，但是同时必须保证写入该地址的内容总是最新的。

在 QSM 中的限制处理机只有在一个读的子操作的开始阶段才知道实际的全局地址，这种限制是有意识地反映 MIMD 机器的读竞争机制。在这种竞争机制中，同一个处理机所做的读操作被放到一个管道中，这样做的好处是可以缓冲读共享存储器的延迟。同样，对于写操作，也采用了这种管道方法。而对于本地计算，由于只占用本地处理机的时间，因此不需要用管道缓冲。

可以把 QSM 模型中的共享存储看成每个全局存储地址的一个队列，新到的读写某一个地址的请求就放到该地址对应的队列中去，每次只能处理一个请求，系统中争论和冲突的最大值可以认为是该队列中的最大延迟。QSM 模型中一个步骤的总开销由该步骤中争论和冲突的最大值、各个处理机所执行的本地操作次数的最大值和各个处理机对共享存储器的读写的最大次数 3 部分组成。为了反映并行机上有限的通信带宽，采用参数 $g>=1$，来表示单位时间内本地指令操作次数与通信次数的比率。

3 各种通信模型的分析与比较

上面所列出的通信模型及它们的扩展模型基本上包括了比较通用的所有的并行计算模型，对它们的设计思想和原理的了解非常有利于我们对新的模型的设计与研究。接下来对上面所说的模型做一个分析和比较。

总的来说，由于 PRAM 模型比较直观、编程简单、符合传统的程序设计概念，因而得到了广泛应用。但是在 PRAM 模型中，并行计算机被理想化了。它假定并行处理机进行的内存存取和计算都是同步操作，而且忽略了同步的开销，效率较低。而在大规模并行计算中，现有的 MIMD 机器一般是异步的，所以要在 MIMD 机器上实现 PRAM 模型是非常困难的，同步代价太高。所以，在 PRAM 上研制的并行算法不能在分布式存储系统上有效执行。而在 BSP 模型，也是把并行计算机简化成可以进行大块同步通信的计算机。也就是说在这种情况下，算法或者程序设计人员必须把算法也设计成与 BSP 模型一致，这在某些情况下，算法不能设计成这种模式的时候就无法使用 BSP 模型。另外在 BSP 模型中，参数 g 是难以确定的，测试这些参数也有一定的难度。因为当集群实际运行的时候，网络中的通信和计算负载往往是不可预知的，这样就导致 g 的值也是不同的。同样，Postal 模型和类 LogP 模型都有着这个问题，当一个集群中的负载不确定的情况下，这些模型都无法实际地反映系统的性能。

下面再对刚才所说的几种模型做一个比较，如图 1 所示。

模型名 (Model)	同步性 (Synchrony)	通信方式 (Communication)	所用的参数 (Parameters)
PRAM	单步 (lock-step)	共享存储 (shared memory)	P
BSP	大块同步 (bulk-synchrony)	消息传递 (messages-passing)	P, g, L
Postal Model	异步 (asynchronous)	消息传递 (messages-passing)	P, l
LogP Model	异步 (asynchronous)	消息传递 (messages-passing)	P, g, l, o
QSM	大块同步 (bulk-synchrony)	共享存储 (shared memory)	P, g

图 1 各个并行计算模型的比较

从图 1 中可以直观地看出，PRAM 模型、QSM 模型都是共享存储式并行计算模型，它们的不同之处在于 PRAM 是单步执行，各子步之间严格地同步，系统的并行粒度低；QSM 是大块同步，在一个步骤之内的各子步之间可以并行执行，系统的并行粒度更高。当 QSM 模型中的 $g=1$，就变成了 QRQW PRAM 模型。

而 BSP 模型、Postal 模型和 LogP 模型都是分布式的消息传递式并行计算模型。BSP 与 QSM 一样都是采取大块同步的同步方式，在一个超步之内各个处理机可以并行执行。而 Postal 模型和 LogP 模型则更进一步加大了并行的粒度，各处理机一直并行执行，只是在特定的时候进行消息传递的网络通信。

虽然 PRAM、QSM 都是共享存储式并行计算模型，这虽然与集群式超级计算机的体系结构不一样，但是它们的设计思想都是值得借鉴的。我们可以把分布式存储系统中的消息传递的通信也抽象成对于共享存储器的读写。而 BSP、Postal 和 LogP 模型更适合于现在的集群式计算机。

总结及进一步研究的方向 通过上面的分析，我们基本可以得出结论：它们都还存在着各种缺点，没有一个模型可以非常精确地刻画不同的计算模型下的网络和系统的性能。

并行计算模型将来的发展趋势很可能由下面两个方面来表现：第一，进一步可能把带宽、延迟、处理机开销等参数抽象出来，用更通用的模型来刻画集群系统的性能，就像集群式系统在带宽和延迟上的性能都比 MPP 系统更差，但是总体性能，也就是实际性能却是更好一样。这说明带宽和延迟并不能真正反映系统的性能。下一代并行计算模型可能会有更高的抽象性。第二，也许将来的并行计算模型会是与具体的计算模式相结合，针对某一个具体的应用来，某一个特殊的模型就能够非常忠实地表现出集群的性能。换句话说，就是有一种模型能够刻画出某一种应用在另一种集群上的性能。这应该是下一代模型的主要发展方向。

最后，由于并行计算模型和并行式计算机的发展是相辅相成的，它们都是由实际应用所驱动的，它们的发展都要取决于新的应用的产生和发展。这也是并行计算模型的一个发展

趋势。

参 考 文 献

1 Anderson T E, Culler D E, Patterson D. A case for NOW (Networks of Workstations). Micro IEEE, 1995, 15(1): 54～64

2 Alexandrov A, Ionescu M, Schauser K E, et al. LogGP: Incorporating Long Messages into the LogP Model-One Step Closer Towards a Realistic Model of Parallel Computation. In: Proc. Seventh Ann ACM Symp Parallel Algorithms and Architectures, 1995. 95～105

3 Moritz C A, Frank M I. LoGPG: Modeling network contention in message-passing programs. Parallel and Distributed Systems, IEEE Transactions on, 2001, 12(4): 404～415

4 Krizanc R, Saarimaki A. Bulk Synchronous Parallel: Practical Experiencewith a Model for Parallel Comuting, Parallel Architectures and Compilation Techniques. In: Proc. of the 1996 Conf., 1996. 208～217

5 Gibbons P B, Matias Y, Ramachandran V. The QRQW PRAM: Accounting for Contention in Parallel Algorithms. In: Proc. of the fifth annual ACM-SIAM symposium on Discrete algorithms, 1994. 638～648

6 Bar-Noy A, Kipnis S. Designing Broadcasting Algorithms in the Postal Model for Message-Passing Systems. In: Proc. of the fourthannual ACM symposium on Parallel algorithms and architectures, 1992. 13～22

7 Culler D E, Karp R M, Patterson D, et al. LogP: Towards a Realistic Model of Parallel Computation. Communications of the ACM, 1996, 39(11): 78～85

8 Ino F, Fujimoto N, Hagihara K. LogGPS: A Paallel Computational Model for Synchronization Analysis. ACM SIGPLAN Notices, 2001(7): 133～142

9 Touyama T, Horiguchi S. Performance Evaluation of Practical Parallel Computation Model LogPQ. In: Proc. of the 1999 International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN '99), 1999. 216～222

10 Moritz C A, Frank M I. LoGPC: Modeling Network Contention in Message-Passing Programs. In: Proc. of the 1998 ACM SIGMETRICS Joint Intl. Conf. on Measurement and modeling of computer systems, 1998. 254～263

11 Gibbons P B. What good are shared-memory models? In: Proc of the 1996 ICPP Workshop on Challenges for, 1996. 103～114

(上接第 111 页)

任给层次结构 H，设 $lts(H)=\{T_1, T_2, \cdots, T_n\}$。我们将采用"all-at-once"方式得到的 LTS 记为 $once(H)$，即：

$once(H)=_{df}(\prod_{i=1}^{n} T_i)\backslash hid(H)$

下一引理说明了 $once(H)$ 也可以通过组合方式计算得到。

引理 1 任给一个 LTS 层次结构 $H=(H_1 \| H_2 \| \cdots \| H_n)\backslash L$，则 $once(H)=(\prod_{i=1}^{n} once(H_i))\backslash L$ 成立。

证明：设 $lts(H_i)=\{T_i^1, T_i^2, \cdots, T_i^{n_i}\}$，记 $\prod lts(H_i)=_{df}\prod_{i=1}^{n} T_i^j$，我们推理如下：

(1) $once(H)=(\prod_{i=1}^{n}\prod lts(H_i))\backslash((\bigcup_{i=1}^{n} hid(H_i))\cup L)$ 由定义

(2) $once(H_i)=(\prod lts(H_i))\backslash hid(H_i)$ 由定义

(3) $(\prod_{i=1}^{n} once(H_i))\backslash L=(\prod_{i=1}^{n}((\prod lts(H_i))\backslash hid(H_i)))\backslash L$ 由(2)

(4) $(\prod_{i=1}^{n}((\prod lts(H_i))\backslash hid(H_i)))\backslash L=(\prod_{i=1}^{n}\prod lts(H_i))\backslash(\bigcup_{i=1}^{n} hid(H_i))\backslash L=(\prod_{i=1}^{n}\prod lts(H_i))\backslash((\bigcup_{i=1}^{n} hid(H_i))\backslash L)$ 由(3)、假设(A)和定理 2

(5) $once(H)=(\prod_{i=1}^{n} once(H_i))\backslash L$ 由(1)和(4)

得证。

下一定理反映了组合可达性分析的基本原理：对于等价语义≈而言，使用 CRA 方法得到的系统模型与采用"all-at-once"方法得到的系统模型是等价的。

定理 3 任给一个 LTS 层次结构 H，公式 $cra(H)\approx once(H)$ 成立。

证明：根据层次结构的定义递归证明如下：

1. 如果 $H=P\backslash L$，其中 P 是一个 LTS，那么 $cra(H)=red(P\backslash L)$、$once(H)=P\backslash L$。因为 $red(P\backslash L)\approx P\backslash L$，所以 $cra(H)\approx once(H)$。

2. 如果 $H=(H_1 \| H_2 \| \cdots \| H_n)\backslash L$，其中 $H_i(i\in[1\cdots n])$ 为 H 的子层次结构且 $cra(H_i)\approx once(H_i)$，令 $H'=_{df}(cra(H_1) \| cra(H_2) \| \cdots \| cra(H_n))\backslash L$，我们推理证明如下：

(1) $cra(H)=red((\prod_{i=1}^{n} cra(H_i))\backslash L)$ 由定义

(2) $cra(H)=red(H'), red(H')\approx H'$ 由定义

(3) $cra(H)=H'$ 由(2)

(4) $once(H)=(\prod_{i=1}^{n} once(H_i))\backslash L$ 由引理 1

(5) $cra(H_i)\approx once(H_i)(i\in[1\cdots n])$ 前提假设

(6) $\prod_{i=1}^{n} cra(H_i)\approx\prod_{i=1}^{n} once(H_i)$ 由(5)和定理 1

(7) $(\prod_{i=1}^{n} cra(H_i))\backslash L\approx(\prod_{i=1}^{n} once(H_i))\backslash L$ 由(6)和定理 1

(8) $H'\approx once(H)$

(9) $cra(H)\approx once(H)$ 由(3)和(8)

综上所述，$cra(H)\approx once(H)$，得证。

结论 本文对标记迁移系统的组合可达性分析的基本原理做了详细分析和形式化描述，证明了相关结论。本文的工作有助于深入理解和阐明组合可达性分析的内部工作原理，从而为基于标记迁移系统的计算机辅助设计和验证提供支持。

参 考 文 献

1 Milner R. A Calculus of Communicating Systems. New York: Springer-Verlag, Inc, 1982

2 Brookes S D, Hoare C A R, Roscoe A W. A theory of communicationg sequential processes. Journal of the ACM (JACM), 1984, 31(3): 560～599

3 Graf S, Steffen B, Lütgen G. Compositional minimization of finite state systems using interface specifications. Formal Aspects of Computing, 1996, 8(5): 607～616

4 Cheung S C, Kramer J. Enhancing compositional reachability analysis with context constraints. In: Proc. of the 1st ACM SIGSOFT Symposium on Foundations of Software Engineering, Los Angeles, California, United States, 1993

5 Sabnani K K, Lapone A M, Uyar M ü. An algorithmic procedure for checking safety properties of protocols. IEEE Transactions on Communications, 1989, 37: 940～948

6 Cheung S C, Kramer J. Checking subsystem safety properties in compositional reachability analysis. In: Proc. of the 18th Intl. Conf. on Software Engineering, Berlin, Germany, 1996

7 Tai K C, Koppol P V. Hierarchy-based incremental analysis of communication protocols. In: Proc. of First Intl. Conf. on Network Protocols, 1993. 318～325

8 Yeh W J, Young M. Compositional reachability analysis using process algebra. In: Proc. of the Symposium on Testing, Analysis, and Verification, Victoria, British Columbia, Canada, 1991

9 Valmari A. Compositionality in state space verification methods. In: Proc. of 17th Intl. Conf. in Application and Theory of Petri Nets (ICATPN'96), Osaka, Japan, 1996