计算机科学1992Vol.19№.1

并发模型的比较研究

肖育东（450052郑州大学计算机科学系）

摘　要

并发性是计算机科学面临的主要挑战之一。近几年来，在这方面已经展开了大量的研究。并发进程模型的重要意义在于，它将提供一个可信赖的通用形式框架及一批新的技术手段，为并行分布式系统的研究、开发打下一个坚实的基础。对模型的深入研究，使我们能更深刻地洞察并发性的实质问题。本文对目前最成功的两个并发模型：CSP和CCS的各主要方面进行较深入的比较研究，搞清楚它们实质的贡献、差异及存在的问题。

并发性乃是计算机科学面临的主要挑战之一，至今已提出各种各样的并发机制以及并发程序设计方法及语言。与顺序程序设计相比，由于并发性的先天复杂性，并发程序设计及其理论和方法，仍是人类尚未充分认识的领域。最早提出的并发模型是Petri网，其研究的丰富成果已广泛用于并发性研究的理论及实践。本文不打算讨论这方面的工作，而是集中于另两个杰出的模型：CSP (Communicating Sequential Processes)[1-3]及CCS (Calculus of Communicating Systems)[4-7]模型，它们分别由C. A. R. Hoare及R. Milner几乎同时分别独立地加以发展。自七十年代以来，在诸多的并发进程模型中，这两个模型是最为成熟，最为系统，也最具成果的模型。下面综合性地比较研究它们的发展、背景、贡献、差异、及应用方面，以便搞清楚并发性问题的实质。

CCS与CSP的基本记号有如下的对应：

CCS	CSP
a·p	a→P
a·p+b·Q	(a→P)|(b→Q)
0	STOP

一、算子的选择

CCS及CSP的基本出发点都是建立一种能反映并发进程计算的通用形式模型，给理论研究及实际应用提供一个牢固的数学框架。但是，二者在构造模型的思路上又明显不同。CSP模型仔细地区分了不同的算子，并以最纯的形式加以引入，确保每一个算子有明确而简单的意义，从而提供一组单纯而清晰的运算规则。结果是CSP具有较多的算子。相对照，CCS主要注重理论上的简单、干净，它引入尽可能少的算子，以包含最大的表达能力。结果必然是一个算子有多方面的意义。下面就几个重要的算子作一比较。

1. 并发合成

CSP中有两种并行合成算子"‖"及"|||"。(P‖Q)表示进程P与Q在(αP∩αQ)中的所有事件必须是同步互锁发生的，而单独属于P(或Q)的事件的发生不作任何时间上的假定。(P|||Q)则允许P与Q的事件以任何相对速度发生，其迹是P的迹与Q的迹的任意交互。"‖"算子可推广到多个进程在其字母表交集上的强制锁定：$\|_{i=1}^{n} P_i$，外部环境不可能破坏这种锁定同步作用。这两种算子的作用机制是清楚而简单的。

CCS中有一个并发合成算子"|"。在概念上它比CSP的"‖"算子远为复杂，它包括了非确定、交互、同步及隐藏（内部化）诸方面。（P|Q）同步，仅当P、Q中分别存在α及其对偶$\overline{\alpha}$时，同步化作用将这一对偶的作用内部化，成为不可见的作用τ，但这种同步作用不是强制的，P与Q仍可以可见地，独立地与外部环境发生作用：

$$a\cdot P\mid\overline{a}\cdot Q=\tau\cdot(P\mid Q)+a\cdot(P\mid\overline{a}\cdot Q)+\overline{a}\cdot(a\cdot P\mid Q).$$

对应地，在CSP中

$$(C\rightarrow P)\parallel(C\rightarrow Q)=C\rightarrow(P\parallel Q)$$

这里同步锁定是严格的，且不存在默认的内部化隐藏作用。

一般地，CCS中

$$a\cdot P\mid b\cdot Q=a\cdot(P\mid b\cdot Q)+b\cdot(a\cdot P\mid Q)$$

这里a，b可以相同。但在CSP中，依据a，b属于不同的字母表及字母表的交集，有不同的作用规则：

$a\rightarrow P\|b\rightarrow Q=(a\rightarrow(P\|b\rightarrow Q))\mid(b\rightarrow(a\rightarrow P\|Q))$，当$a\in\alpha P-\alpha Q$，$b\in\alpha Q-\alpha P$；

$a\rightarrow P\|b\rightarrow Q=a\rightarrow(P\|b\rightarrow Q)$，当$a\in(\alpha P-\alpha Q)$，$b\in(\alpha P\cap\alpha Q)$；

$a\rightarrow P\|b\rightarrow Q=b\rightarrow(a\rightarrow P\|Q)$，当$a\in(\alpha P\cap\alpha Q)$，$b\in(\alpha Q-\alpha P)$；

$a\rightarrow P\|b\rightarrow Q=STOP$，当$\{a,b\}\subseteq(\alpha P\cap\alpha Q)\wedge a\neq b$。

2. 删除与限制

在CCS中有限制算子"\"，在CSP中有删除算子"\"，它们书写形式相同，但有着不同的意义。CCS的限制算子是简单地阻止一些事件的发生，而CSP的删除算子是使一些事件的发生成为外部不可见的内部事件。

在CCS中，当存在α，$\overline{\alpha}$对偶时自动进行事件的内部化。但是，若存在二个以上的事件可能配成作用对偶，则选择哪一对事件作配偶，或者不配偶，是不确定的：

$$a\cdot P\mid a\cdot Q\mid\overline{a}\cdot R=\tau\cdot(P\mid a\cdot Q\mid R)+\tau\cdot(a\cdot P\mid Q\mid R)+a\cdot(P\mid a\cdot Q\mid\overline{a}\cdot R)+a\cdot(a\cdot P\mid Q\mid\overline{a}\cdot R)+\overline{a}\cdot(a\cdot P\mid a\cdot Q\mid R).$$

CCS使用限制算子阻止一些事件发生，这样增加了选择的强制性，限制了一部分不确定选择的可能性，如：

$$(a\cdot P\mid a\cdot Q\mid\overline{a}\cdot R)\setminus\{a\}=\tau\cdot((P\mid a\cdot Q\mid R)\setminus\{a\})+\tau\cdot((a\cdot P\mid Q\mid R)\setminus\{a\})$$

它使a·P及a·Q不再能以端口a独立地对外部发生作用，仅限于共享进程$\overline{a}\cdot R$。

二、非确定性

CSP中，删除运算使一些事件不可见地发生，这可能使原来可观察可控制的事件被隐藏，对于外部观察者来讲变为不可见的非确定事件。形如$(x:B\rightarrow P(x))$的进程如果删除其初始事件集B，则导致形如$(\sqcap_{x:B}P(x))$的不确定选择。更令人遗憾的是，若被删除的事件集在进程中构成一个不可见的内部无限循环事件序列，则进程将可能忙于不可见的无休止循环，它在任何有限时间内对外部作用将不作出反应。这称为进程"发散"。

对于CCS，不确定性是并发合成算子的固有性质，即"|"运算将引入不确定选择（前已给出例子）。仅当限定

$$(L(P)\cap L(Q)=\varnothing)\wedge(\overline{L(P)}\cap L(Q)=\varnothing)$$

的情况下，（P|Q）能保留合成的确定性。其中L表示标号集。CCS不直接表现CSP中所述那种发散情形，因为其内部作用由τ来表示，我们仍可以跟踪τ的作用踪迹，因此它能区别不同的发散，而不是（象CSP中那样）把所有的发散都归为特殊进程CHAOS。但是CCS中却不能区别一个发散进程以及一个行为像发散而其实不发散的进程。

CSP中的算子对（‖，\）及CCS中的算子对（|，\）都与内部化及不确定性有关。概念上，由它们导致的非确定性或发散性是不可避免的。即，不存在一种通用的方法来避开这一问题。这一点恰反映了并发性固有的复杂性及难处理性。但是，在实践中，人

们决不愿意（不允许！）设计一个非确定的、难以预测的、不可控制的系统。我们只有针对具体设计问题，分析发生不确定性的可能原因，通过一些设计技术，使它不发生。

三、关于等价分类

CCS是一个研究并发模型的等价关系族的一般性框架，重点在强等价关系及观察等价关系。强等价类似于动态迁移过程中所保留的一种同构关系（称为强双模拟）。观察等价允许忽略不可见的内部作用，它的一个重要性质是$P\approx\tau\cdot P$，是观察等价能力的来源。但可惜，观察等价性对于求和运算"+"不保留，即由$P\approx\tau\cdot P$并不能得出$Q+P\approx Q+\tau\cdot P$。我们知道，等价类对于运算的封闭性是很重要的。为此，CCS中定义了观察等同（记为$=$）类。观察等同要求：若$P\approx Q$，则P与Q的任意派生$\tau^*\alpha\tau^*$之后（设其达到P'与Q'），仍有$P'\approx Q'$。换句话说，P，Q是观察等同的，当且仅当P，Q是观察等价的，而且对于任何可观察作用之后仍是观察等价的。观察等同对于所有算子都是保留的，它是比观察等价更强的一类等价关系。如$P\approx\tau\cdot P$，但一般地$P\neq\tau\cdot P$，$P\mid\tau\cdot Q\approx P\mid Q$，但$P\mid\tau\cdot Q\neq P\mid Q$。

观察等同与观察等价有如下关系：

$$P\approx Q\Longleftrightarrow (P=Q\vee P=\tau\cdot Q\vee\tau\cdot P=Q)$$

CCS中这一套概念还可以依迁移关系的深度重新建立起来，从而从等价的动态性质上来洞察这一套概念及其相互之间的关系。CCS的这一成绩堪称天才的杰作。

CSP中的等价关系是建立在失效语义模型上的，称为失效等价（记为$\approx_f$）。失效等价在CSP上的全部运算（除"／"之外）都是保留的。失效等价的最大贡献是：它是能够区分（非确定）死锁进程与非死锁进程的最弱等价。因此，在并发程序设计中用于进程性质的证明有着重要的实用价值。

对于确定性进程，失效等价模型可以简化成迹等价模型，即仅由〈字母表，迹〉对，就可以唯一地表示一个确定性进程。但把迹等价类用于非确定情况，其主要缺点是不能区别死锁与非死锁进程。迹等价是更弱的一类等价。

综上所述，从等价类的强弱，依次可列出：

- 强等价（$\sim$）
- 观察等同（$=$）
- 观察等价（$\approx$）
- 失效等价（$\approx_f$）
- 迹等价（$\approx_t$）

它们之间具有真包含关系。但从其标准定义严格讲，观察等价与失效等价是不可比的，对于系统的发散性，CSP的失效等价有着明显的分界，而观察等价却不如此。对观察等价定义加以修正之后，可以使其为真包含关系。直观上讲，观察等价对于分支情形比失效等价更为敏感，例如：$A\overset{\text{def}}{=}a\cdot(b\cdot c\cdot O+b\cdot d\cdot O)$，$B\overset{\text{def}}{=}a\cdot b\cdot c\cdot O+a\cdot b\cdot d\cdot O$，则$A\approx B$，而$A\not\approx_f B$，原因是对于失效等价，分配律的限制形式$\alpha\cdot(\beta\cdot P+\beta\cdot Q)\approx_f\alpha\cdot\beta\cdot P+\alpha\cdot\beta\cdot Q$成立，但对于观察等价却不成立。

失效等价与迹等价的不同，不在于其迹，而在于其拒绝集与发散集，因为迹是失效集的一个组分。例如，令

$$P\overset{\text{def}}{=}a\cdot(b\cdot O+c\cdot O),$$

$$Q\overset{\text{def}}{=}a\cdot b\cdot O+a\cdot c\cdot O,\quad (b\neq c)$$

则 $P\not\approx_f Q$，　但　$P\approx_t Q$。

但是对于确定性进程，观察等价与迹等价是一致的，因而也与失效等价，这三者是一致的。

四、说明

一个模型中允许使用逻辑的形式说明进程的性质，并进行推理，这是模型能力的重要标志之一。对于并发进程模型，是指在事件及作用的意义下建立一套进程逻辑，能把系统或事件与某个"说明"联系起来。换句话说，系统或事件满足某个"说明"，是指它具有某

些“性质”。

CCS在标号迁移系统

$$(P,\ Act,\ \{\xrightarrow{\alpha}:\alpha\in Act\})$$

之下定义了PL逻辑，建立了类似Hoare逻辑的一组断言规则，并研究了在强等价意义下迁移作用的断言保持性。它使用迁移作用的结构链讨论二事件之间的不类似程度。PL逻辑是部分正确性的证明逻辑。由于这一逻辑建立在迁移关系而不是组合子（算子）上，因此，它演绎的是说明的结构而不是程序的结构。

CSP使用“满足性”指派一个进程的说明，形如$P\ sat\ S$，它说明的是程序性结构。演绎的是进程行为中的不变性条件。这种说明的逻辑运算与组合子运算之间有一种对应，因此，它允许从子结构的说明来合成较大的说明。这种逻辑更接近于直觉主义的演绎系统，它不必论证一个程序不满足某说明。

五、语义问题

近些年来已经对于CCS及CSP模型及其变种的语义方面展开了深入的研究。

由G. Plotkin的结构操作语义[9]发展起来的迁移语义方法用于CCS，把全部计算化归为基本计算[8]，其中使用的迁移归纳是在推理树上依深度归纳的特殊情况。迁移语义作为CCS的操作语义，是CCS的语义基础。Scott及Strachey[10]建立在论域上的指称语义，以及由此进化出的幂域理论（以指称不确定性行为的语义），是目前广泛接受的语义方法。可惜，目前已发现指称语义与操作语义之间的匹配是不完全的。这种意义失配的情形也出现在并发进程模型之中。[11]

CSP的基本语义模型是失效语义。已经知道，失效模型可以由一个公理系统所产生的项代数来构造。项代数存在完全的证明系统。[12]

对于CCS，CSP的无结构，或受限条件下的结构（或其变种），往往能得到优美的形式系统。CCS中若不允许递归及无限和，可得到有限事件公理系统；如果不允许静态组合子，则得到有限状态事件上的公理系统，这二个公理系统都是完全的，可判定的。Milner引入了与CCS密切相关的一类语言，称为同步计算（SCCS），它适宜于表示同步进程的计算。

一个并发模型可以有多种语义解释，它们从不同的侧面刻画模型的性质，但可能又存在一定的局限或缺陷，不能满意地处理模型的所有方面。我们确实要有一种判据，以比较这些解释，并拒绝那些不恰当的解释。当一种语义解释与模型的操作语义不匹配时，从模型的目的来看，应以操作语义为准。为判别一个语义模型符合的程度，我们可以使用观察等价的概念表述为：二个行为表达式有相同的语义解释，当且仅当在所有的环境中它们对于外部观察者是不可区别的。

进程到底是什么样的数学对象还没有取得一致的意见，搞清楚各种并发语义方法的内部联系是一个重要的任务，这有助于改善我们对于并发性的理解。

六、可实现性

CSP使用最纯的原始概念来刻画并发进程的性质，而不在于高度抽象。因此，CSP有较好的实现条件。按CSP模型已开发了实验性语言OCCAM并研究了它的语义，除了给出失效模型上的指称之外，还用一种“混合”的方法解决关于共享变变量等问题[13]。

CCS企图建立一种更为广泛、更为抽象的并发原型，主要用于研究并发进程的数学性质，但实现技术上存在较多的问题，如：可以产生无限多个系统成分，可以有任意的互连通讯，可以不确定地选择程序的执行，等。CCS理论本身不能恰当地处理公平性问题，不能容易地提供“依名字调用”的机制，不能严格地区别发散等。这些问题使得CCS作为一种语言目前尚不能全面有效地实现。当前更为关注的是，CCS到底能给计算机实践带来什么新方法、新概念，新结果。在一

个新基础上的实现才可能是突破性的，正象λ-演算带来的LISP及函数式程序设计语言那样。

七、其他并发模型

CCS与CSP共同的基本假设是：基于"观察"，站在外部观察者的立场来研究并发系统；基于"握手式"通讯，通讯在进程之间无缓冲，无延迟，是直接的，这类似于电话系统。这两点假设与现代程序设计方法学有较好的相容性，因此，许多现存的成功程序设计概念，方法，手段都可以在CCS或CSP模型上重建或发展。

Petri网模型的出发点则很不相同，它从事件及事件发生的条件入手，研究行为的因果关系，从而构成系统的静态及动态图式。Petri网的研究及应用已取得较大的较果，其中较多地是研究系统的各种数学特性（可达到性，安全性等），较少用于程序设计的直接支撑。目前，对一些问题已经从不同的模型框架得到了实质相同的结果。但Petri网与CCS，CSP的关系是不清楚的。

另外还应提及Hewitt的Actor模型。它是一个非同步模型，具有很强的表达能力。它与CCS，CSP的不同点是：①一个Actor所发送的消息将在有限时间之后达到其目标Actor，且对消息的到达次序不作任何假设和限制。这类似于邮政系统而不是电话系统。这一放松使得通讯的递归成为可能。②允许Actor动态地产生。因此，它支持一些优美而有力的程序设计方法。这两点使得对于Actor系统的理论分析带来新的困难。Actor与CCS，CSP的关系看起来应当是很密切的，抑或把CCS或CSP的条件放松或扩充，看看能告诉我们一些什么？这些问题目前尚不清楚。

参考文献

[1] Brookes, S.D., Hoare, C. A. R., and Roscoe, A. W., A Theory of Communicating Sequential Processes, JACM, vol.30, pp560—599, 1984

[2] Hoare, C. A. R., Communicating Sequential Processes, CACM, vol.21, pp666—677, 1978

[3] Hoare, C. A. R., Communicating Sequential Processes, Prentice Hall, 1985

[4] Milner, R., Communicaton and Concurrency, Prentice Hall, 1989

[5] Milner, R., A Calculus of Communicating Systems, LNCS, Vol. 92, Springer-Verlag, 1980

[6] Milner, R.,A Calculus of Communicating Systems, Report ECS-LFCS-86-7,Compnter Science Department, Univ. Edinburgh, 1986

[7] Milner, R., Lectures on A Calculus for Communicating Systems, in Control Flow and Data Flow(ed. M. Broy), pp205—228, Springer - Verlag, 1985

[8] Plotkin,G.D.,A Structural Approach to Operational Semantics, Report DA-IMI-FN-19, Computer Science Dept., Arhus Univ., Denmark, 1981

[9] Milner, R., Calculi for Synchrony and Asynchrony, J. Theoretical Computer Science, Vol. 25, pp267—310, 1983

[10] Scott, D., and Strachey, C., Towards a Mathematical Semantics for Computer Languages, Proc. Symp. on Computers and Automata, Microwave Res, Inst. Symposia Series, Vol.21, Polytechnic Inst. of Brooklyn, 1972

[11] Milne, G., and Milner, R., Concurrent Processes and Their Syntax, JACM, Vol.26, 2, 1979

[12] Hennessy, M., and de Nicala, R., Testing Equivalences for Processes, Proc. ICALP, 1983, LNCS154(1983)

[13] Roscoe, A. W.,Denotational Semantics for OCCAM, LNCS, 197. Seminar on Concurrency, Springer- Verlag, 1984, pp306—329