并行程序设计语言Occam及其运行机Transputer

刘晓华（中国科学院数学研究所，北京）

摘　要

本文讨论并发程序设计语言Occam及其运行微机Transputer的主要特性。Occam用一组并行操作且通过通道通信的进程来描述外部世界，而Transputer作为并行处理系统的一个积木块，用Occam作为其互连的设计体系，将进程映射到Transputer（网络）部件上，自动实现并行或并发处理（单机或Transputer网络）。

一、Occam

1.1 语言综述

Occam是一个处于实用阶段的并发程序设计语言，与它相依的机器是微计算机Transputer（INMOS）。该语言用一组并发进程来描述一个系统，这些进程之间及进程与外设之间通过点对点通道通信。通常Occam程序有三种基本进程：

赋值进程——将表达式值赋给一个变量，其语法形式为v:=expression；

输入进程——通过通道c给变量赋值，c?v；

输出进程——通过通道c输出表达式之值，c!expression。

这些基本进程按照一定的方式组合起来，可以形成丰富的Occam程序结构：串行结构（SEQ），并行结构（PAR）和选择结构（ALT）；组成这些结构的进程之间的逻辑关系分别是顺序，并发和择一的关系，而执行关系分别是顺序执行，并发执行和先准备好的先执行。这些结构本身也是一个进程，因而可作为另一个结构的一个组成元；这些结构相互无限制的嵌套构成了丰富的Occam程序层次结构，大大提高了语言的任务描述能力。

进程的逻辑结构由三部分组成：进程自身，进程的输入和进程的输出（图1）。Occam语言里，进程从输入通道输入信息，从输出通道传送结果；进程之间需要交换信息时，一个进程的输出通道和另一个进程的输入通道链接，当这两个进程都准备就绪时，通信就可顺利进行。通信完成以后，进程继续各自的动作；否则，一个进程等待另一个进程发送信息或接收信息，直到两者通信可顺利进行为止。

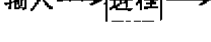

图1. 进程的逻辑结构

和普通的程序设计语言一样，Occam语言也有条件（IF），循环（While），重复，开关（Case）等控制结构。其中条件，循环，开关语句和普通的计算机语言类似，但重复控制结构稍有不同，重复符（Replicator）有SEQ，PAR，IF和ALT四种，它们重复执行组元（进程）规定的次数。重复结构的一般形式是

REP index=base FOR count

这里*REP*只能是*SEQ*，*PAR*，*IF*或*ALT*四者之一。例如若P_i $(0 \leqslant i < 5)$ 是一组进程，则

```
PAR i=0 FOR 5
  P[i]
```

构成了一组（5个）结构类似的并发进程。

由于Occam是支持并发处理的程序设计语言，所以和顺序型语言相比，增加了一般顺序型语言所没有的语言成份，即在语言级增加了通信命令（进程），允许用户定义自己的通信协议，指明两个并发进程通信所用的通道，通信严格遵循通信协议的规定。通信命令和通信协议的一般形式是

c? v　从通道c输入并把值送给变量v，

c! e　从通道c输出表达式e之值；

和

```
CHAN OF protocol name
```

基本协议（protocol）是INT，BYTE。复合协议可像C语言里的宏定义一样首先形式化定义一个协议名。例如

```
PROTOCOL Message IS BYTE; INT:
CHAN OF Message chan1:
```

其中chan1是类型为Message的通道变量。通道说明像变量一样说明，它表明通道传送数据的数据类型和数据结构。接收信息的变量必须和传送的数据类型和结构保持严格一致。

并发程序行为的测不准性，导致了新的语言结构——选择（Alternative）结构的引入，用它可实现程序运行不确定时的一种可能的选择，但这并不影响程序运行的正确性。

此外，Occam还有两个特殊的进程SKIP和STOP；在程序开发阶段，它们可替代一个尚未开发的过程。启动进程SKIP后，什么也不做就退出该进程；启动进程STOP后，该进程不继续也不结束，外部行为类似死锁现象。有时这两个进程在程序设计中是必要的。例如，一个进程表示有旅客乘坐飞机时飞机才起飞，(测试时)如果没有旅客乘坐该机，则启动SKIP，让这个进程正常终止。进行程序设计时，处理错误的进程可用进程STOP代替。

进程STOP还有另外一层含义。Occam程序有逻辑错误，如IF语句的条件不充分，含有该IF语句的进程在运行条件进程（IF语句）时，若当前条件不包括在IF语句所列的条件之中，则该进程就会产生像启动进程STOP的效果一样。这个被毁坏（broken）进程会因与其它进程有信息交换而使整个（并行处理）系统被毁坏或发生死锁。

Occam语言也允许用户定义过程（关键字PROC）和函数（关键字FUNCTION），取决于过程或函数定义时的形参说明，参数的传递方式既可以传值也可以传结果。由于本文不打算专门介绍Occam程序设计语言，关于语言的细节在此不详述。

1.2 并行处理

Occam是基于并行执行的概念而设计的，它提供并发进程之间的通信和自动同步机制。并行结构（PAR）将可并行执行的部分组织成一个结构，运行时这几部分就可并发执行，它们之间通过同步通信而相互制约。Transputer的微码处理器（程序）能使任意数目的并发进程一起执行，分享处理器时间。

顺序型程序只有一个起点和一个终点，而并发程序可有多个起点和多个终点。Occam程序的并行执行地点可设置在程序的任一可执行地点。

1.3 理论基础

并发程序设计语言Occam主要的基本思想源于C.A.R.Hoare的通信顺序进程(CSP)〔Hoare，1978〕。在CSP里，Hoare提出了输入输出作为程序设计的基本原语和通信顺序进程的并行组织作为基本的程序结构方法，给出了进程通信的模型——用输入输出命令联结两个并发进程；利用Dijkstra的卫式命令（guarded command）〔Dijkstra，1975〕来描述并发程序运行的不确定性。

Occam的输入输出进程基于CSP的输入输出命令，并行结构以CSP的程序结构方法为基础，并发进程的通信和同步都是CSP的；并发程序的不确定性控制也是采用CSP的控制方法。可以说，Occam是第一个严格按照Hoare的CSP通信理论模型来设计实现的并发程序设计语言。

二、微机Transputer

2.1 概貌

英国INMOS公司开发的微型计算机Transputer面向并行处理，1985年9月推出第一个支持Occam并行处理模型的IMS T414 32位Transputer。

Transputer是一种VLSI器件，它包括处理器，存贮器（存贮处理器执行的程序）和直接与其它Transputer相连的数个通信接口（communication links）。处理器有16位和32位，分别可带16位和32位外部存贮器接口。处理器和存贮器都融合在同一集成电路器件上。每个Transputer有自己的局部存贮器，能用于单处理器系统，或者多个Transputer构成Transputer网络，建立高性能的并行系统。此外，它还有特殊的接口，如链适配器，使Transputer和其它外设（如IBM PC机）相连。

Transputer可实现器件的内部通信和器件之间的通信，后者较前者速度慢得多，但都是利用硬件优势实现点点通信。器件之间的通信速度因使用单向信号线（每根线连接两个器件）而被优化，但如果多个器件共享总线连接，则通信速度就会受到限制。

2.2 体系结构

组成Transputer的基本器件是一系列可编程的VLSI元件。Transputer的逻辑结构规定了一个互连的Transputer系统是如何设计和编程的，其物理结构规定了这些Transputer是如何实现互连和控制的。一种Transputer的体系结构如图2所示：

系统服务器包括Transputer操作的初始化和维持操作需要的所有逻辑，还有错误处理和分析功能。

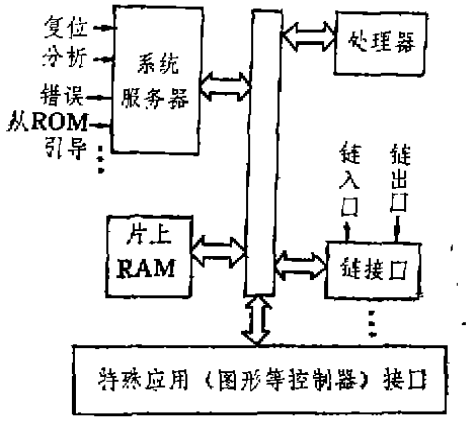

图2 Transputer体系结构

链接口是和其它Transputer连接实现通信的接口，有一个链入口和一个链出口。

2.3 系统原理

Transputer可看作一种硬件积木，利用一系列Transputer（积木）可构筑一个具有高性能行为的并行系统。这种硬件支持(Transputer）器件间通信和器件内通信，也支持并发处理。

系统根据内连的进程集设计，一个进程是一个软件积木块，一个独立设计单元，用点对点通道实现与其它进程的同步通信，其内部设计是隐蔽的，被视作为一个“黑盒”。

Transputer体系结构采用点对点通信链。每一个Transputer有一个或数个标准的INMOS通信接口（链接口），每个通信接口可与其它的Transputer的通信接口连接，能构成任意大小和拓扑结构的Transputer网络。标准的INMOS通信链由通信接口和连接两个Transputer间通信接口的两根单向的既传送数据又传送控制信息的信号线组成，使得Transputer系列能用直接的点对点连接构成网络，无需外部逻辑。

两个Transputer之间用两条单向信号线，将一个Transputer上的链接口接到另一个Transputer上的链接口，数据（信息）就在单向信号线上串行传输；通信链的这两条信号线可提供两个Occam通道，一个方向一个。

为确保通信同步，系统要确认每一条信息，通信链协议使同步通信顺利有效正确实施（图3）。协议保证字节按任意顺序传输，使不同字长的Transputer能相互连接实现通信；信息按字节顺序传送，每个字节在下一个字节被传送之前必须被确认。字节按如下

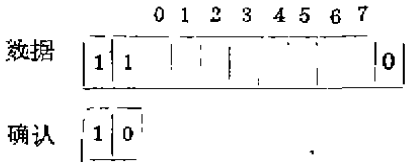

图3 链协议和确认格式

顺序传输：

起始位→第一至八位数据位→终止位

接收信息的Transputer有一个单字节缓冲器，保证信息不丢失。通信链上发送的每一个数据字节都在与当前信号线相连的链入口被确认。数据字节传送完后，发送者等待确认信息直到收到确认为止。确认表明：进程能接收确认的字节和接收者能接收下一个字节。发送者仅在信息的最后一个字节被确认后才重新调度发送进程（当信息需用外部通道传送时，处理器把发送信息的进程从调度表中移走，并把传送信息的任务交给链接口，信息传送完后，链接口使处理器把刚才移走的那个发送信息的进程加在调度表中）。

每条信号线的数据字节和确认信息是重复的，确认信息在数据字节一开始接收就能传出，从而传输是连续的，数据字节间无延迟。

在同一个Transputer上执行的两个进程，其间的通道用存贮器内的单字实现，而通信通过存贮器移动数据来实现。

通信时，输入进程和输出进程哪一个先准备好无关紧要，通信链协议要求先准备好的输入（输出）进程等待另一个与之通信的输出（输入）进程准备就绪。

三、Occam和Transputer的有机结合

3.1 结合

并发程序设计语言Occam与微机Transputer几乎是同时开发的，而且Transputer支持Occam的编程模型。用Occam编写的程序可在单一的Transputer上运行，Transpute在其并发进程间实现处理器分时，并通过存贮器内移动数据实现通道通信；它系也可在列Transputer构成的网络上运行，每一个Transputer执行分配给它的进程，并发进程通过Transputer间的通信链实现外部通道通信。针对Occam的各种并发和通信模型，所有的Transputer都包含专用的指令和硬件，保证以最高的性能实现这些模型。例如，选择结构（ALT）的组成元是些进程（在Occam里），这些进程要么等待从几个通道接收信息，要么在等待一规定时间后从通道接收信息，这些动作都由Transputer提供直接的有效的指令，实现对ALT结构的逻辑控制。

3.2 映象

Occam程序是一组（并发）进程的集合，Transputer能将这些并发进程进行适当映象，使Occam并发程序可在一个Transputer或Transputer网上运行。图4和图5给出了三个并发进程P1，P2，P3分别在一个Transputer和三个Transputer上的映象。

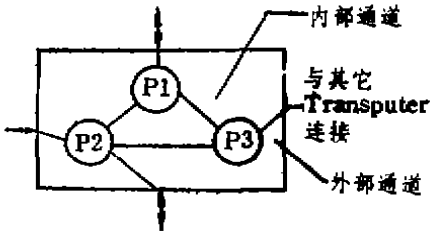

图4 进程在一个Transputer上的映象

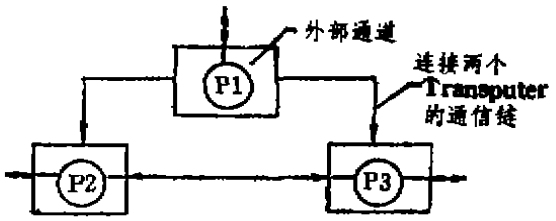

图5 进程在三个Transputer上的映象

为了给多个Transputer正确配置Occam程序，Transputer开发系统提供了必要的工

具。这种配置可以使某些并发进程具有高优先级。高优先级的进程先执行，低优先级的进程只有在优先级比它高的所有进程不能执行或执行完后才能执行（条件允许的话）。这种配置不影响程序的逻辑性能。

Occam程序的逻辑性能指计算结果不受实时效应的影响，不会因进程映射到处理器的方式不同而改变，或者因处理和通信速度而改变。由于这种特性，在单机或多机(Transputer网）上开发Occam程序并无差异，单机上的Occam程序仍然可用一组并发进程描述事件，除了明显的通道交互操作外，进程完全独立地使用自己的变量操作。

四、结束语

自从Dijkstra提出结构化程序设计概念以后，结构化程序设计给予了人们有益的帮助。然而，计算机科学家仍在不断努力研究结构化程序设计的新概念，“结构化”在一些语言中也不断得到体现和更新。Occam语言一提出来就体现了较强的结构层次。

用结构化程序设计语言编写的程序，条理清晰，结构鲜明，可读性好，而且具有良好的分布、并行或多重处理特性，易于在计算机网络上分布，在多处理器上并行。Occam和Transputer的结合正丰富和具体化了这些优点。

由于Transputer自身的硬件特性，Transputer可作为并行处理系统的一个积木块使用，用Occam作为其互连的设计体系。利用Tranrputer可构造各种新颖的体系结构，从分时计算机，多重处理机，直到相互通信的计算机网络，适合于模拟，机器人控制，图象合成以及数字信号处理等。

程序的正确性一直是计算机科学家致力研究的课题，筑得了一些突破。为保证程序设计语言的逻辑正确性，需要一种数学模型（数学语义）刻画语言的精确，无歧义，安全和稳定。六十年代后期，人们发现数学语义的重要性。R.W.Floyd提出语义是一组正确的证明规则；C.A.R.Hoare研究通信顺序进程，提出了刻画各种运算的重要特性的代数法则；Robin Milner对并发性数学模型进行了深刻的研究，提出了通信系统演算理论，解决了CSP早期设计中许多无法解决的问题，如并行命令可否相互嵌套？若是，则能否用递归过程并行地调用自己？理论上可否用输出命令构造卫式等等。由于Occam严格基于CSP，因此Occam语言语义遵循上面的数学模型。

本文在写作过程中，部份地参考了清华大学王鼎兴教授和他的课题组提供的资料，同时得到了陆汝钤教授的悉心指导，他对本文提出了修改意见，并对一些词的译法作了斟酌，作者特致衷心的谢意。

（接第50页）

加强了GDASS系统的灵活应用能力。

（3）为了体现决策者在决策过程中的作用，系统提供了决策者干预决策过程的接口。决策者通用属性效用函数的建立，可以表示出在一定的决策态势下决策者对风险的态度，决策者可利用决策模型控制知识库来控制决策过程，提高系统解决实际问题的灵活性。

随着效用函数知识库和决策模型控制知识库的不断完善，会进一步提高GDASS系统的性能。

（4）GDASS系统针对不同情形，提供了三种推理形式，决策者依据具体的决策领域，灵活地选择推理方式，就能达到较好的决策效果。

（5）GDASS系统提供了决策结果分析子系统，使决策者能够对决策结果的有效性、可靠性以及适应范围有一个清晰的了解。此外，数据库管理子系统也都加强了系统的灵活运用能力。

总之，将人工智能技术引入DSS，不仅有助于提高决策者的决策质量，而且也有助于扩大决策者作出决策能力的范围。

本系统目前正在ST-286机上做模拟试验，这里主要给出GDASS系统的模式或框架，并给出主要子系统的功能及模块功能而略去一些设计细节。从模拟试验证明该系统的总体设计思想从预处理到GDASS的建立是行之有效的，可以在实践开发中与同行进行商讨。（参考文献略）