第8章　规格说明阶段

一个规格说明文档必须满足客户和开发者两方面相互矛盾的需求。一方面，对一个可能不是计算机专家的客户来说，此类文档必须清晰而且具备可理解性。毕竟，客户要为产品付酬金，除非客户相信他真正理解了新产品的蓝图，否则客户很有可能会决定不授权产品开发，或叫别的开发组织来开发。

另一方面，规格说明文档必须是完备的和详细的，因为它是负责拟定设计的设计组唯一可用的信息源。即使客户已认定所有的需求在需求阶段已经精确地定义了，但是，如果规格说明文档中包含了遗漏、矛盾、模糊不清等错误，则设计中的这些错误将不可避免地被带到实现中。因此，需要使用一些技术来表示目标产品，这些技术应能以一种非技术的描述形式来表示产品，以便于理解；而且应足够地精确，以保证在开发周期的最后，交给客户的产品是无缺陷的。规格说明技术是本章和下一章的主题。本章的重点放在结构化规格说明技术，而下一章则致力于面向对象的分析。

8.1　规格说明文档

实际上，每个规格说明文档都包含有产品必须满足的约束条件。任何产品总有交付的最后期限；另一个常见的规定是“本产品应和现有的产品安装在一起并行地运行”，直到用户认定这个新产品确实满足了规格说明文档的各个方面。其他的约束可能包括可移植性，即产品应能在相同操作系统不同的硬件上或各种不同的操作系统上运行。可靠性可能是另一个约束条件。如果产品用于一个特护病房里照看病人，那么能在一天24小时里不停地运行就显得极为重要了。快速响应时间可能另外一个需求；在这个方面一个典型的约束可能是“类型4的95%需求应在0.25秒内做出响应”。许多的响应时间约束以概率性术语来表达，因为响应时间依赖于计算机的当前负荷。相反，所谓的硬性实时约束则以绝对性术语来表达。例如，如果一个软件“只能在95%的时间里能在0.25秒内通知战斗机飞行员正在遭到一枚导弹的攻击”，这样的软件是毫无用处的，该产品必须满足100%的时间约束。

规格说明文档的一个重要组成部分是验收标准集。从客户和开发商的观点来看，讲清楚证明该产品真正满足它的规格文档的测试条件，而且开发商也确实做到了，这是很重要的。其中的一些验收标准可能就是约束条件的重述，然而其他的标准可能针对其他的问题。例如，客户可能给开发商提供一些该产品需要处理的数据的说明。于是相应的验收标准可能是：产品能正确地处理这些类型的数据，并过滤掉与这些类型不一致的(即错误的)数据。

规格说明文档是客户和开发商之间的一项合同。它精确地表明了产品必须做什么，以及对产品的约束条件。一旦开发组完全地理解了问题的实质，就能提出可能的解决策略。“解决策略”是构造产品的大体方法，例如，一个产品的一个可能的解决策略是使用联机数据库；另一个策略可能是使用传统的平面文件，并通过夜间的批处理运行抽取所需的信息。在决定解决策略时，在不考虑在规格说明文档中约束条件的情况下提出策略，是一个好办法。然后可以根据约束条件来评价各种解决策略，并做一些必要的修改。有很多方法能够谓词一个特定的策略是否能满足客户的约束条件。一个显而易见的方法是通过原型，这对于解决和用户界面及定时约束有关的问题来说，不失为一个好技术；这一点已在前面3.5节中讲述过。其他一些谓词约束

是否被满足的技术包括仿真[Brately,Fox和Schrage,1987]和分析网络模型[Jain,1991]。

在这个过程中，大量的解决策略不断地提出而后又被扔掉。保存一份所有被抛弃的策略和被抛弃原因的书面记录是很重要的。这将帮助开发组回忆是否这个方案已经遇到过。但更重要的是，在维护阶段的增进维护过程中，通常会尝试提出一个新的、不明智的策略，这是非常危险的。保存一个在开发过程中某种方案为何被抛弃的记录，对维护阶段是很有帮助的。

到现在，开发组已经确定了一个或多个能满足约束的可能的解决方案。现在进行一个两步的决定过程。首先，是否要建议客户实现计算机化，如果是，则应采用哪个可行的解决策略。对第一个问题应根据成本-效益分析(4.2节)来做出决策。其次，如果客户同意让项目继续进行下去，那么客户必须向开发组提供最优标准，例如客户希望的最小总成本和最大的投资回报。于是，开发商向客户建议哪一种方案能最好地满足这些最优标准。

8.2 非形式化规格说明

在许多开发项目中，规格说明文档是用英语或其它的自然语言(如中文)书写的。一个规格说明文档的典型段落可能是：

“BV.4.2.5。如果本月的销售额低于标准销售额，那么需要打印一个报表，除非目标销售额和实际销售额之差小于上个月的目标销售额和实际销售额之差的一半，或者除非目标销售额和实际销售额之差低于5%”。

上面段落的背景如下：一个零售连锁店的管理部门对每个商店规定了月度的目标销售额。如果某店未达到这个目标，则打印一个报表。考虑到下面的情况：假如1月份某店的销售目标额为10万美元，但实际销售额只有6.4万美元，也就是说，低于目标36%。在这种情况下，必须打印一份报表。现在更进一步假定，二月份目标额为12万美元，而实际销售额只有10万美元，低于目标销售额16.7%，虽然销售额低于目标数字，但二月份的百分比16.7%低于前一个月百分比差额36%的一半。管理部门相信销售情况有所好转，于是不用打印报表了。现在假定三月份的目标销售额又为10万美元，而实际销售额为9.8万美元，仅仅低于目标销售额2%。因为百分比差额很小，低于5%，所以也不用打印报表。

仔细地重读前一段的规格说明文档，说明它和零售管理部门实际要求的差异。段落BV.4.2.5提到了“目标销售额和实际销售额之差”，而没有提到百分比之差。一月份的销售差额为3.6万美元，而二月份销售差额为2万美元。管理部门要求的差额从36%下降16.7%，低于一月份差额的一半。可是，实际销售额之差从3.6万美元下降到2万美元。而2万美元大于3.6万美元的一半。这样如果开发组根据说明文档如实地实现，将会打印一份报表，而不管这是不是管理部门所希望的。该段最后一句提到“差额...低于5%”，意味着为5%差额，在该段落中的其他部分并未提到百分比。

规格说明文档中存在着大量缺陷。首先，它忽略了客户的需求，其次存在着许多模糊性，如最后的“差……5%”或“差额……5千美元”，或是别的什么？而且，其文档风格也不直观。这段文档的意思是：“如果发生某些情况，就打印报告；然而，如果发生别的情况，就不打印报告。而且如果发生其他情况，也不打印报告”。如果规格说明文档只描述在何时打印报表，这样可能更清晰。总之，BV.4.2.5段并不是一个关于怎样写一个规格说明的好范例。

实际上，BV.4.2.5段是虚构的，但不幸的是，太多的规格说明文档就是这样。也许读者认为这个例子不公平，而且这种问题不可能发生，因为规格说明文档毕竟是专业的文档人员写的。为了反驳种论点，下面对第5章的例子进行研究。

实例研究：文本处理

让我们回忆一下5.2.2节。在1969年，Naur发表一篇关于正确性证明的论文。他通过文本处理问题解释了他的技术。使用他的技术，Naur构造了一个ALGOL过程解决这个问题，并且非形式化地证明了他的过程的正确性。一位评论家在评论Naur的论文时指出了该过程中的一个错误[Leavenworh,1970]。其后，London检测到三个其他的错误，并且发布了该过程的一个纠正版，然后形式化地证明了它的正确性[London,1971]。之后，Goodenough和Gerhart又找到London未发现的三个错误[Goodenough and Gerhart,1975]。在评论家London、Goodenough和Gerhart所发现的7个错误中，有2个是关于规格说明文档的。例如，Naur的规格说明文档没有说明，如果输入包括两个连续的相邻分隔符(空格或换行字符)时将发生什么。为此，Goodenough和Gerhart产生了一套新的规格说明文档。这个文档大约是Naur文档的4倍，这在5.5.2节中已经给出。p77

在1985年，Meyer写了一篇关于形式化规格说明文档技术的文章[Meyer,1985]。其中心议题是用英语之类的自然语言书写的规格说明文档往往存在矛盾、模糊和忽略等问题。他推荐使用数学技术来形式化地表达规格说明文档。Meyer在Goodenough和Gerhart规格说明文档发现了12个错误，于是他开发了一套数学规格说明技术来纠正所有的问题。然后，Meyer意译了他的数学规格说明，形成了英语规格说明文档。在作者看来，Meyer的英语规格说明文档中也包含一个错误。Meyer在他的论文中指出，如果每行的最大字符数是10，而输入的如果是WHO WHAT WHEN，于是，按照Naur文档及Goodenough和Gerhart文档，存在着两种有效的输出，即WHO WHAT在第一行,而WHEN在第二行，或WHO在第一行，而WHAT WHEN在第二行。实际上，Meyer的英语规格说明也存在这种二义性。

重要的是，Goodenough和Gerhart的规格说明文档是以最大的细心构造出来的。首先，他们构造这些文档的原因是为了修正Naur的规格说明文档。其次，Goodenough和Gerhart的论文经历了两个版本，第一个是在会议上发布的，第二个发表在期刊上[Goodenough and Gerhart,1975]。最后，Goodenough和Gerhart都是软件工程方面的专家，尤其在规格说明文档方面。因此，如果说这两个专家在时间很充足的情况下精心设计的这个文档仍然具有Meyer所发现的12个错误，那么一个普通的计算机人员在时间压力下怎么可能生产出无缺陷的规格说明文档呢？更糟糕的是，一个文本处理问题能用25～30行代码完成，而现实世界中的产品是由几十万甚至上百万行代码组成的。

显然，自然语言并不是一个很好的产品描述方法。本章介绍了其它一些较好的替代的方法，介绍的顺序是从非形式化技术到形式化技术。

8.3 结构化系统分析

将图形运用于软件的规格说明在70年代是一个重要的技术。DeMarco技术[DeMarco,1978]、Gane和Sarsen技术[Gane and Sarsen,1979]、Yourdon和Constantine技术[Yourdon and Constantine,1979]，这三种使用了图形的技术在当时非常流行。这三种技术都很好，而且本质上是等价的。因为Gane和Sarsen技术的表示法将在9.3.3节面向对象的分析中使用，所以在这里我们将介绍这种方法。

为了更好地理解这个技术，我们先来看下面的例子。

Sally的软件商店

Sally的软件商店从各供应商处购买软件，并将其卖给大众。Sally库存那些流行的软件包，并根据需要订购别的软件。Sally将生意做到了大学、公司和个人。Sally的软件商店经营情况

很好。每月有300个软件包的成交量，平均零售价格为250美元。尽管商务运行正常，但有人建议Sally实现计算机化管理。那么，他将如何做呢?

上述问题的描述是不够的，应描述为：哪些商务功能(即入帐、出帐和库存)应该计算机化呢？甚至这还不够，还有：系统是批处理方式还是联机方式？需要使用内部计算机还是外购计算机？但即使问题已进一步精化，仍然错过了一些基本问题：Sally实现计算机化管理的目的是什么?

只有知道了Sally的目标后，分析才能继续下去。例如，如果她希望实现计算机化的目的是为了销售软件，那么，她需要一个带有声光效果的内部系统，以展现计算机的可能功能。

本例假定Sally希望通过计算机来赚取更多的钱，这并无太多的帮助。但很明显，通过成本-效益分析能判定她的三部分生意中哪一个或哪些部分需要计算化。

许多标准方法的主要危险在于，一个人总是喜欢先提出一个解决方案，如使用LimeIII型计算机、380MB硬盘和一个激光打印机，但后来却发现有问题。相反，Gane和Sarsen技术用9步来分析客户的需要[Gane and Sarsen,1979]。一个很重要的一点是这9步中的许多步都采用了逐步求精技术；这一点将在介绍这一技术时展现。

在确定了Sally的需求之后，结构化分析系统的第一步是确定逻辑数据流，这和物理数据不同(即发生了什么，而不是怎样发生的)。这可通过画数据流图(DFD)来完成。数据流图使用如图8-1所示的4种基本符号(Gane和Sarsen的表示法与DeMarco[Demarco,1978]、Yourdon和Constantine[Yourdon and Constantine,1979]的表示法类似，但不完全相同)。

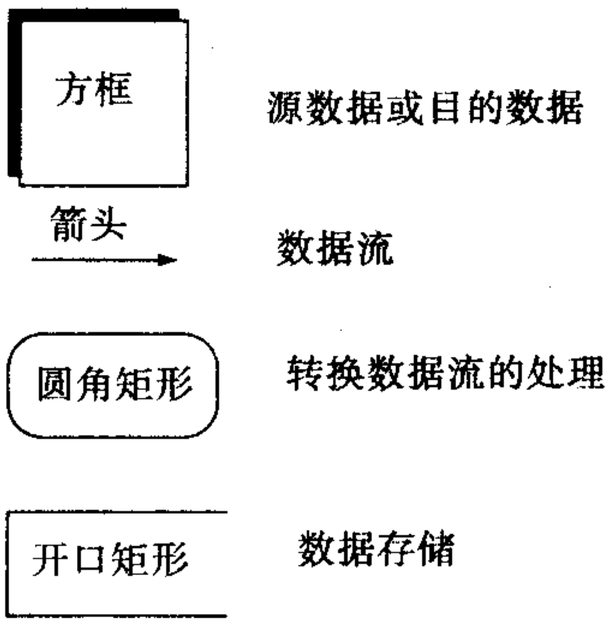

图8-1　Gane和Sarsen的结构化系统分析的符号

步骤1. 画数据流图。

数据流图是逻辑数据流所有方面的图形表示，所以一定包含多于7±2个元素，即任何实际产品的数据流图可能都很大。因此，设计数据流图必须逐步求精。上述例子的第一步求精如图8-2所示。

逻辑数据流的图表有很多解释，两个可能的实现如下：

实现1. 数据存储“软件包数据”由在货架上排列的900盒磁盘和桌子抽屉里的一些目录组成。数据存储“顾客数据”由橡皮筋箍着的5×7卡片和一张超期未付款的顾客列表。处理“处理订单”是Sally寻找货架上合适的软件包，如有必要，可查阅目录，接着查找那张5×7卡片，检查顾客的名字是否在违约者清单上。这种实现总体上是手工的，与Sally处理当前商务的方法一致。

实现2. 数据存储“软件包数据”和“顾客数据”作为计算机文件，“处理订单”操作是Sally在终端上键入顾客的名字和软件包名字。此种实现与所有信息可联机得到的完全计算机化的解决方案一致。

图8-2的数据流图不仅表达了前面两种实现， 而且实际上表达了无穷的可能性。数据流图的关键是表示信息的流动，Sally的顾客实际需要的软件包对于流动来说并不重要。

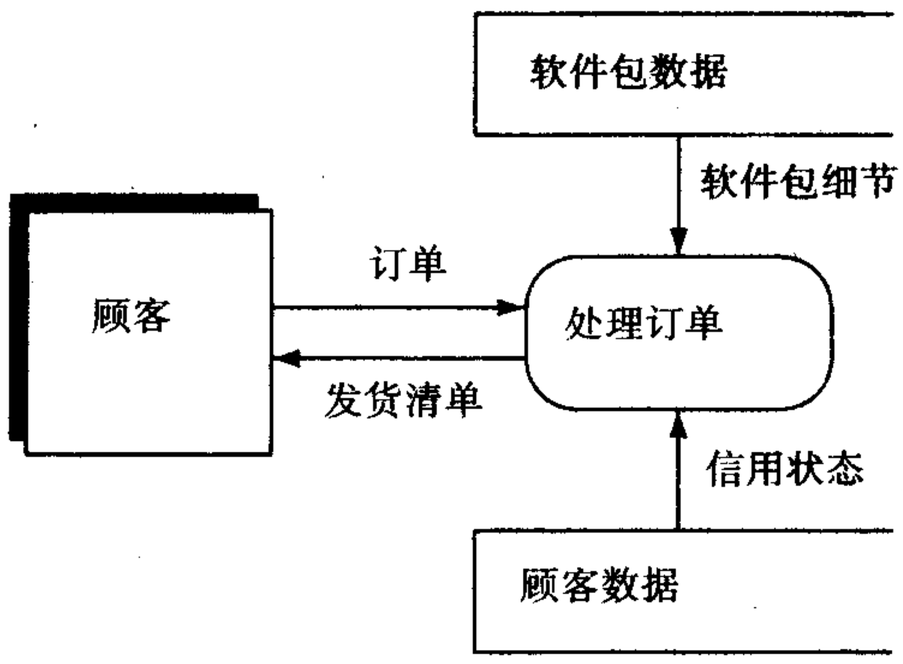

图8-2 数据流图：第一步求精

现在开始对数据流图进行求精。第二步求精如图8-3所示。现在当顾客需要一个软件包而Sally手头上没有时，那个软件包的细节将放入“待决订单”中。这些“待决订单”可能是一个计算机文件，但在这阶段可能只是一叠用纸记录的材料。计算机或者Sally可以每天扫描那些订单，并且，如果向一个供应商订购的订单足够多，就批量订购。而且，如果一个订单已经等待了5个工作日，则不管有多少软件正等着从相关的供应商处订购，先要购买这张订单上的软件包。当软件包从供应商那里到达时，这个数据流图并未显示这个数据的逻辑流；也未显示财务功能，如入帐和出帐。这些将在第三步求精中加入。

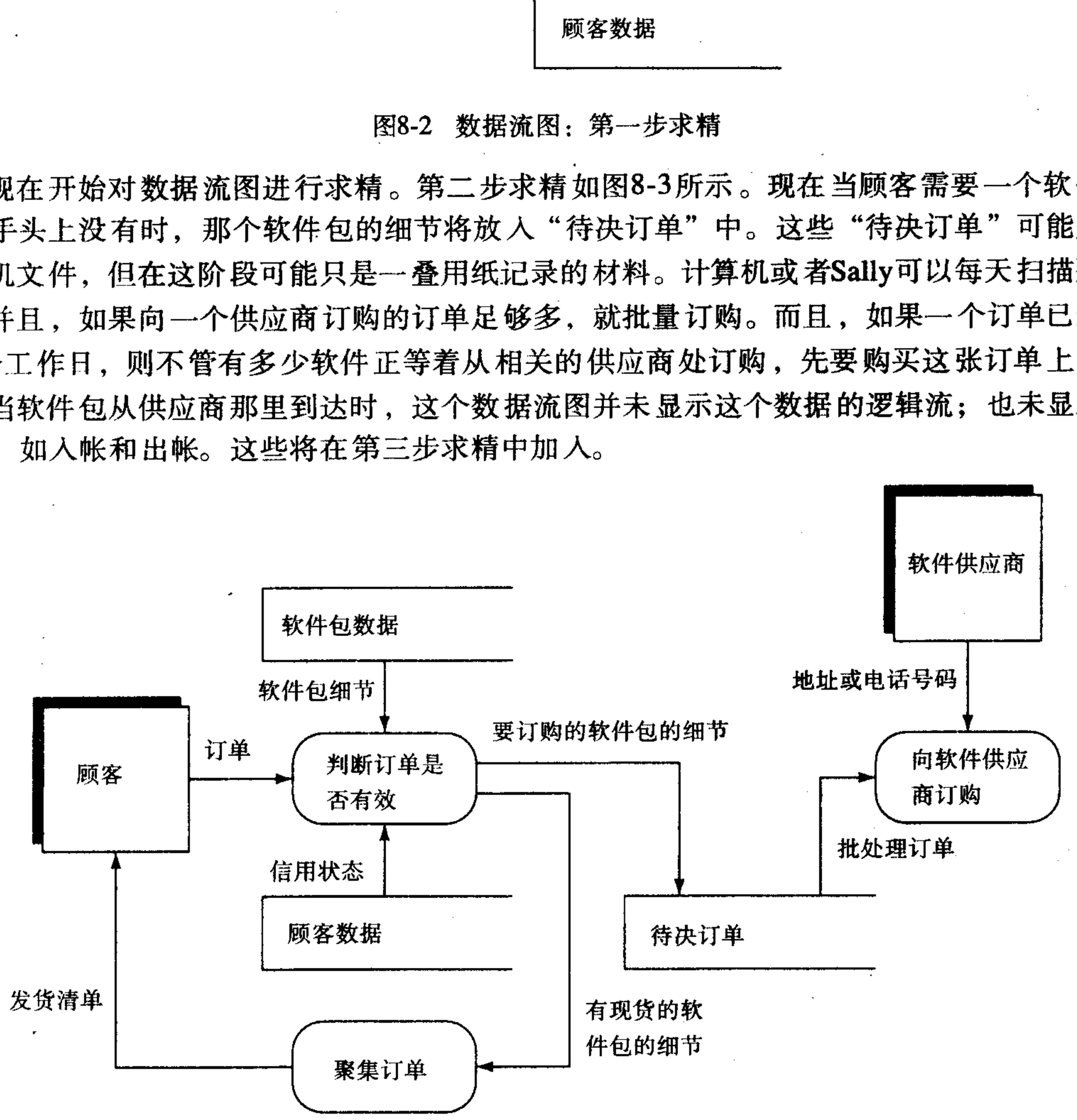

图8-3 数据流图：第二步求精

因为数据流图已经变得很大了，所以在第3步求精图仅有一部分显示在图8-4中。数据流图的其他部分与出入帐及软件供应商有关。最终的数据流图可能更大，可能会扩大到6页。但Sally理解它将很容易；在她确信这个数据流图精确地表达了她的商务数据逻辑流之后，她将签署它。

当然，一个大的产品的数据流图将更大。在到达某一点后，只用一个数据流图将是不切实际的，此时需要一个数据流图层次结构。在某一级的数据流图将被扩展成一个完整的低级数据流图。这种方法经常由于源数据和目的数据的放置问题而带来困难。例如，一个特定处理P反映在L级上，并在L+1级上扩展。处理P的源数据和目的数据的正确位置是在L+1级上，因为这些数据将在这一级上使用。但对一个客户，即使在阅读数据流图方面很有经验，也经常不能正确地理解L级的数据流图，因为处理P相关的源数据和目的数据似乎丢失了。为了防止这种困惑，有时可通过画出正确的数据流图，然后修改它，将源数据和目的数据往上移一级或几级，以消除客户的这种困惑。

△ 步骤2. 决定哪些部分需要计算机化和怎样计算机化(批处理或联机处理)。

对将什么进行自动化的选择通常取决于客户打算花多少钱。显然，将整个操作计算机化最好，但费用问题会阻止这样做。为决定哪部分需要自动化，必须利用成本－效益分析对实现各个部分计算机化的各种可能方案进行分析。例如，对于数据流图的每个部分，必须决定这组操作应以批处理方式还是联机方式执行。在需要处理大批数据和需要严密控制的情况下，批处理方式较好；而在处理量小和使用内部微机的情况下，联机处理方式似乎较好。回到上述的例子，第一个方案是以批处理方式自动处理出帐，而用联机方式处理订单的有效性检查。第二个方案是自动化所有的处理，其中软件供应商的发货票据可使用联机处理方式，也使用可批处理方式，其它操作都是联机方式。关键是数据流图符合前面提到的各种可能性。这种符合不是在规格说明阶段决定怎样解决问题，这些事要等到设计阶段才做。

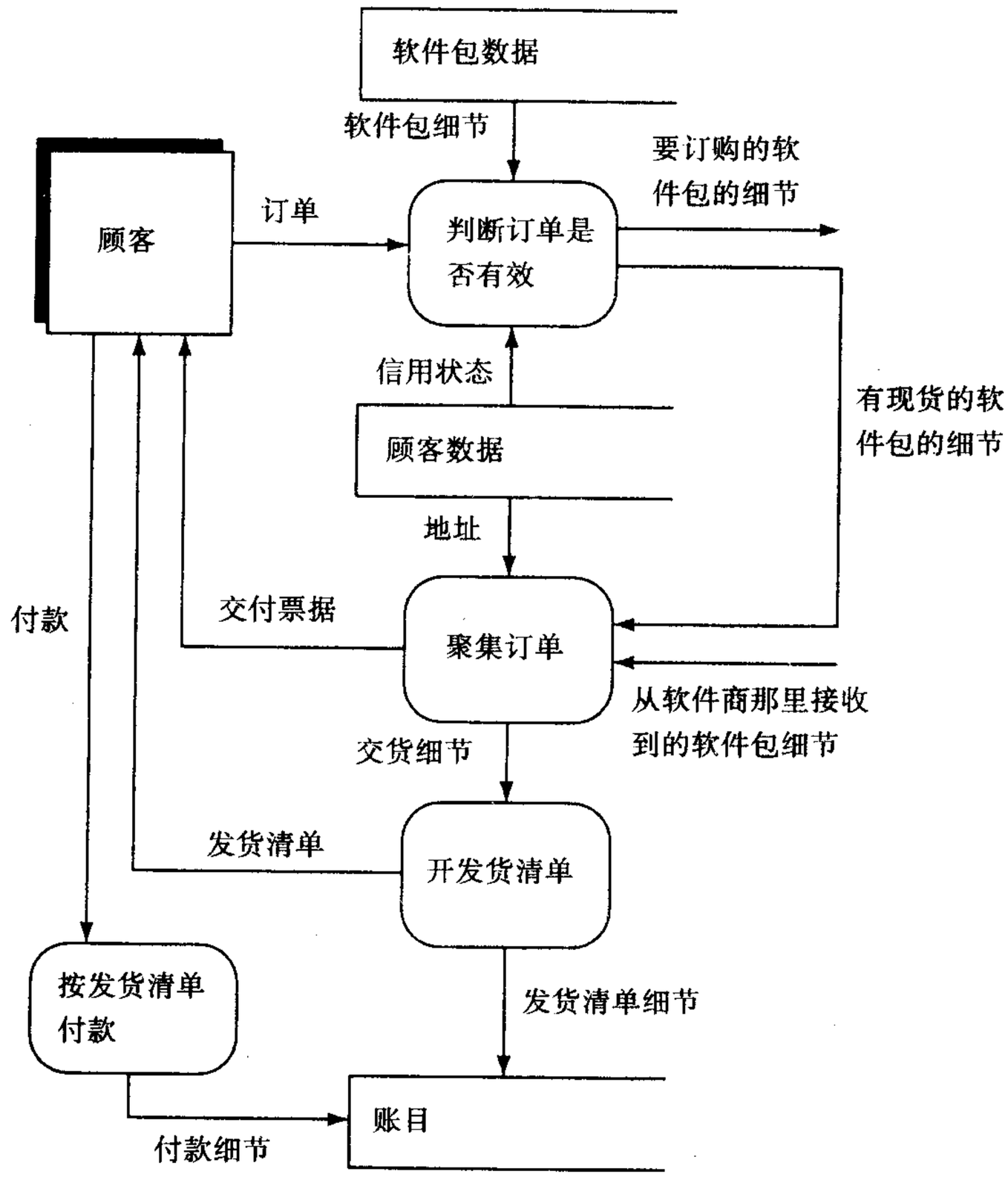

图8-4　数据流图：第3步求精

Gane和Sarsen技术的下面三步是对数据流(箭头)、处理(圆角矩形)和数据存储(开口矩形)求精。

步骤3. 放入数据流细节。

首先，决定哪些数据项必须进入各种数据流。接着，逐步求精每个数据流。

在这个例子中，有一个数据流“订单”，它可细化为：

“订单”：

“订单标识”

“顾客细节”

“软件包细节”

现在进一步细化“订单”的每个部分。在一个大产品的情况下，数据字典(4.3.1节)保存了所有数据元素。

步骤4. 定义处理逻辑

既然已经确定了产品中的数据元素，现在到了研究每个处理该做什么的时候了。假如例子中有一个处理“给教育部门打折扣”。Sally必须向软件开发商提供她给教育部门打折扣的细节。例如，少于等于4件给10%的折扣，多于4件则给15%的折扣。为对付用自然语言表示规格说明文档的困难，这个处理应该从英语转化为一个判定树，如图8-5所示。

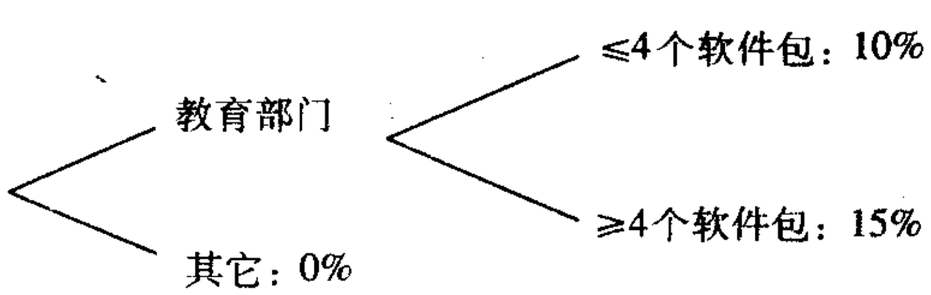

图8-5 描述Sally软件商店教育部门折扣政策的判定树

判定树使得检查是否考虑到了所有的可能性变得容易，尤其在更复杂的情况下，如图8-6所示的例子。从图8-6中立即明显地看出，校友在端区的座位价格没有指定。另一个表达这个过程的好方法是使用判定表[pollack,Hicks,and Harrison,1971]。判定表有一个判定树没有的好处就是,判定表的内容可利用CASE工具自动进入计算机，从而不需要为产品的那部分编代码。

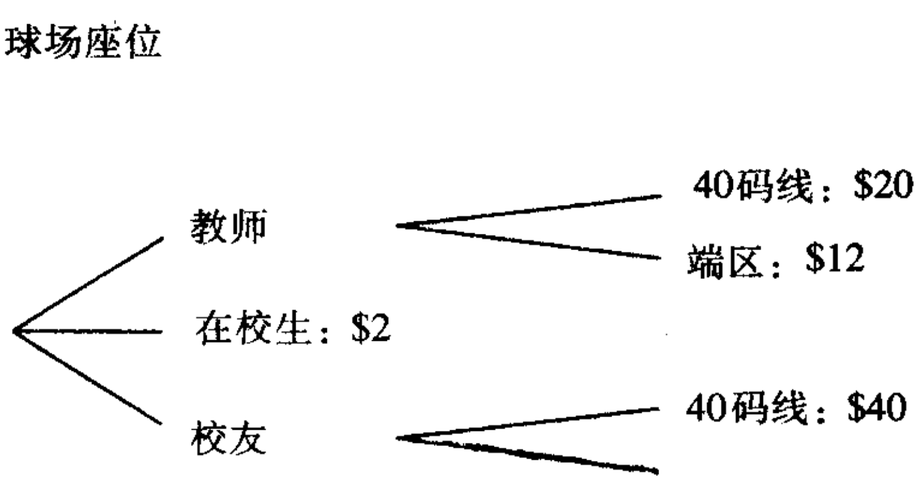

图8-6 描述大学足球比赛座位价格的判定树

步骤5. 定义数据存储。

在这个阶段，必须定义每个存储的确切内容及其表示法(格式)。这样，如果产品用COBOL语言实现，这些信息需要降至pic级；如果使用Ada语言，则需要指定digits和delta。而且，必须指出哪些需要实时访问。

实时访问问题依赖于对产品实施什么样的查询。例如，假定在例子中通过联机方式来谓词

是否能订货。一个客户可以通过名字(“你有Lotus-1-2-3存货吗？”)，或通过功能(“你有什么计算软件包”)，或通过计算机类型(“你有686机上使用的新软件吗？”)来订购软件包，但很少通过价格(“你有值149美元的软件吗”？)来订购软件包。因此，实时访问软件包数据要通过名字、功能、计算机类型进行，如图8-7中的实时访问流图(DIAD)所示。

步骤6. 定义物理资源。

开发商在知道有哪些联机需求和每个元素的表示法之后，就能据此做出关于部件因素的决定。此外，对每个文件必须指定：文件名、组织结构(排序、索引等)、存储介质和记录(字段级)。如果要使用一个数据库管理系统，则每个表的相关信息也要在这里指定。

步骤7. 确定输入／输出规格说明。

必须指定输入格式，即使没有详细的布局，至少也要指定要输入哪些部分。输入屏幕要近似地确定。打印输出格式也要尽可能详细地指定，至少要估计输出的长度。

步骤8. 确定有关的数值。

现在必须计算出第9步确定硬件需求所需的有关数值数据，包括：输入量(每日或每小时)、打印报表的频率、每个报表的最后期限、CPU和大容量存储器之间传输的每一类记录的大小和数量以及每个文件的大小。

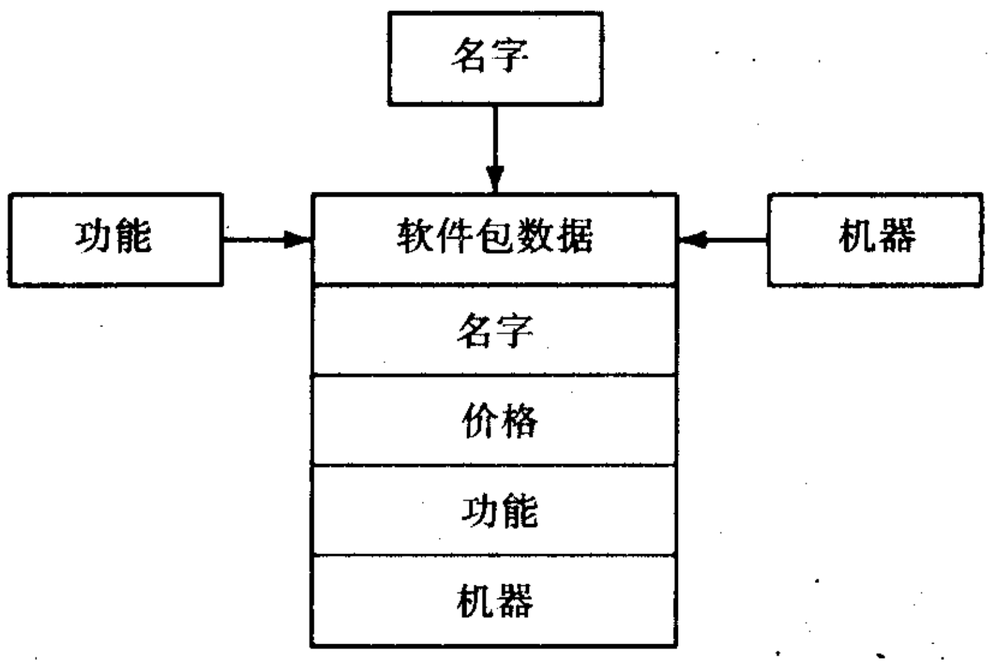

图8-7　软件包数据的数据实时访问图表(DIAD)

步骤9. 确定硬件需求。

从步骤8计算出的磁盘文件的大小信息，就可以计算出所需的存储容量。此外，还要确定备份所需的的大容量存储需求。根据以上得出的输入数据容量信息，可以确定出这方面的需求。由于打印报表的行数和频率已知，就能够指定输出设备。如果客户已经有了一些硬件资源，则可以确定这些硬件是否已足够，或还要不要购买和租借。但是，如果客户没有适合的硬件，则要给出需要哪些硬件的建议，以及这些硬件应是购买还是租借。

确定硬件需求是Gane和Sarsen规格说明技术的第9步，也是最后一步，这样前面这个例子就算完成了。在得到客户的认可以后，最终的规格说明文档就可交给设计组，使软件开发过程继续进行下去。

尽管Gane和Sarsen付出了很大的努力，他们仍然没有解决所有的问题。例如，这种技术不能确定响应时间，输入／输出通道的信息量最多只能大略地估计出。而且，CPU的大小和频率也不能精确地估计。这些是Gane和Sarsen技术的明显缺陷。公平地说，实际上任何规格说明技术或设计技术都存在缺陷。不过，无论是否能得到精确的信息，在规格说明的最后都必须决定硬件需求。尽管如此，这种状况在以数学方法进行规格说明的方法提出之前已经算是最好的了。在这之前，在软件开发过程的一开始，往往关于硬件的决定就已做好了。Gane和 Sarsen技术已

经给产品的规格说明方面带来了很大的进步。尽管Gane和Sarsen 以及大多数规格说明技术的作者们基本上都忽略了将时间作为一个变量，但不能由此否定这些技术的引入给软件工业带来的好处。

8.4 其他的半形式化技术

Gane和Sarsen技术明显地比用自然语言书写的规格说明文档更正规，但同时又没有下面将要讨论的一些技术正规，如Petri网(8.7节)和Z(8.8节)。Dart和他的合作者将规格说明和设计技术划分为非形式化、半形式化和形式化三类[Dart,Ellison, Feiler and Habermann,1987]。根据这个分类标准，Gane和Sarsen的标准化系统分析属于半形式化技术，而本节中提到的其他两种技术则属于形式化技术。

本书引入结构化系统分析技术有两方面原因。首先，它使用很广泛，很有可能读者正在某个正在使用结构化系统分析的公司内工作。其次，数据流图被用于第9章的分析中。但是，仍有很多半形式化技术。你可查阅关于多种软件规格说明和设计方面的国际工作组进行的各种活动的成果。因为篇幅关系，在这里只给出一些知名技术的简单描述。

PSL/PSA[Teichroew and Herskey,1977] 是一个对信息处理产品进行规格说明的计算机辅助技术，其名字来源于该技术的两个组成部分，即用来描述产品的问题描述语言(PSL，problem statement language)和用于将PSL描述输入计算机并生成需求报告的问题描述分析器(PSA，problem statement analyzer)。PSL/PSA得到了广泛的使用，特别是用于为产品编制文档。

SADT[Ross,1985]包括两个相互关联的部分，一个是方框箭头图表语言，称为结构化分析(SA，structural analysis)，另一个是设计技术(DT，design technique)，因此叫做SADT。通过逐步精化可将SADT进一步扩展，使之可利用Gane和Sarsen技术；也可以通过有意识的努力，使之遵从Miller准则。正如Ross所说，“所有值得说的任何事都必须用6个或更少的块来表达”[Ross,1985]。SADT已经在各种各样的产品中取得了成功，特别是在一些复杂的大规模项目中使用非常成功。和许多其他类似的半形式化技术一样，它不太适用于实时系统。

另一方面，软件需求工程方法(SREM，software requirements engineering method，读作“shrem”)可明确地说明某个特定的行为所发生的条件[Alford,1985]。因为这个原因，SREM对实时系统的规格说明尤其有用，而且已扩展到分布式系统领域。SREM由许多部分组成。其中RSL是一个规格说明语言。REVS是一套工具集，它能完成多种与规格说明有关的任务，如将RSL规格说明转换成自动化数据库，自动地检查数据流的一致性(确保一个数据项在赋值之前不被使用)，以及从规格说明中产生模拟器，用来保证规格说明的正确性。此外，SREM还有一个叫做DCDS的设计技术，DCDS即分布式计算设计系统(distributed computing design system)。

SREM的强大来自于该技术的基础模型是一个有穷状态机FSM(finite state machine)。有穷状态机将在8.6节描述。由于SREM是以这个形式化模型为基础的，因此，使它有可能完成前面提到的一致性检查，而且能在给定各个部件性能的条件下验证产品作为一个整体所必须满足的约束条件。美国空军使用SREM说明了两个C^3I(command、control、communications和intelligence)系统[Scheffer,Stone,and Rzepka,1985]。虽然SREM在规格说明阶段已被证明很有效，但REVS工具集在开发周期的后期阶段就不那么有用了。

8.5 实体关系模型

结构化系统分析的重点在于产品的行为，而不是数据。产品的数据也要模型化，这当然是

正确的，但与行为相比则是次要的。相反，实体关系模型(ERM，entity- relationship modeling)是对产品进行规格说明的一种半形式化的面向数据的技术。它被广泛用于数据库的规格说明。而在这个应用领域，重点在于数据。当然，要通过行为来访问数据，数据库要以某种合适的方式加以组织，以使访问时间最短。这种情况下，施加在数据上的行为没有数据重要。

实体关系模型不是一门新技术，从1976年以来它就被广泛使用[Chen,1976,1983]。现在，它又有了新的用途，实体关系模型已成为面向对象分析中的一个成份(参见第9章)。

一个简单的实体关系图如图8-8所示。这个图是作者、传记和读者之间关系的模型。实体有三个，分别是作者、传记、读者。最上面的关系即写作，反映了作者写一个传记的事实。这是一个一对多关系，因为一个作者能写多部传记；这通过作者框旁边的1和传记框旁边的n表示。实体关系图也显示了在传记和读者之间的两个关系。两个都是一对多关系，左边的关系基于一个读者可以阅读多部传记这一事实。类似地，右边那个关系则表示一个读者可以拥有多部传记。将两个关系分开的原因是一个读者能阅读一部传记，但不拥有它，而一个读者也可以购买多部传记但不阅读它。

下一个例子取自供应商和他们供应的部件的关系。图8-9显示了部件和供应商之间的多对多关系。也就是说，一个供应商可供应多个部件；反之，一个部件也能从多个供应商那里获得。这个多对多关系通过供应商旁边的m和部件旁边的n来反映。

更复杂的关系也是可能的，例如在图8-10中，一个部件又可视作由许多零件组成。于是，可能存在多对多对多关系。考虑在图8-10中显示的三个实体：供应商、部件和项目。一个特定部件可由几个供应商供货，这取决于项目，给一个特定项目提供的各个部件可以来自不同的供应商。在这种情况下，必须建立一个多对多对多关系模型。

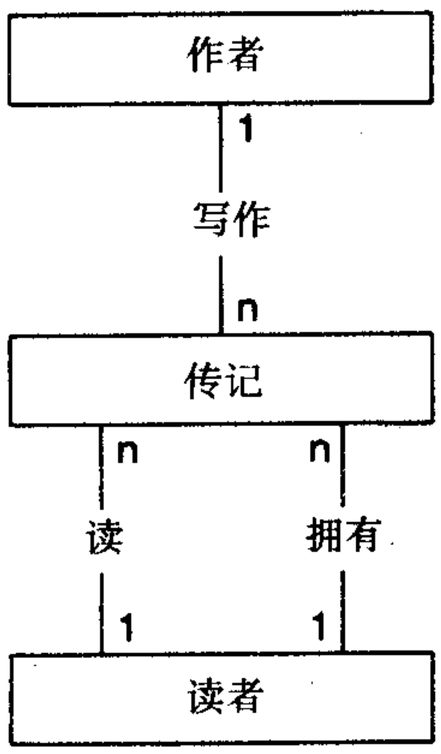

图8-8　简单的实体关系图

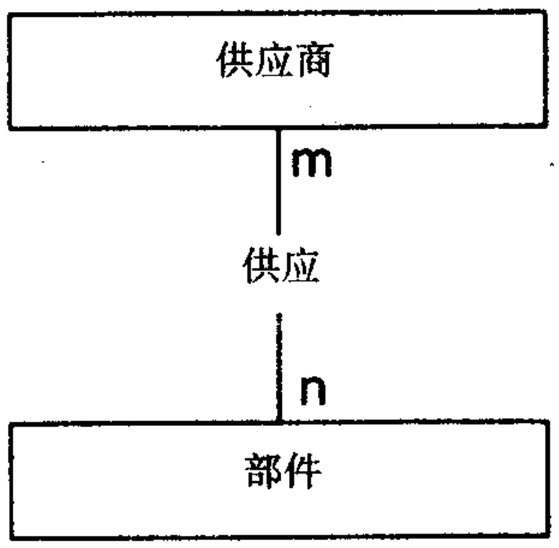

图8-9　多对多实体关系图

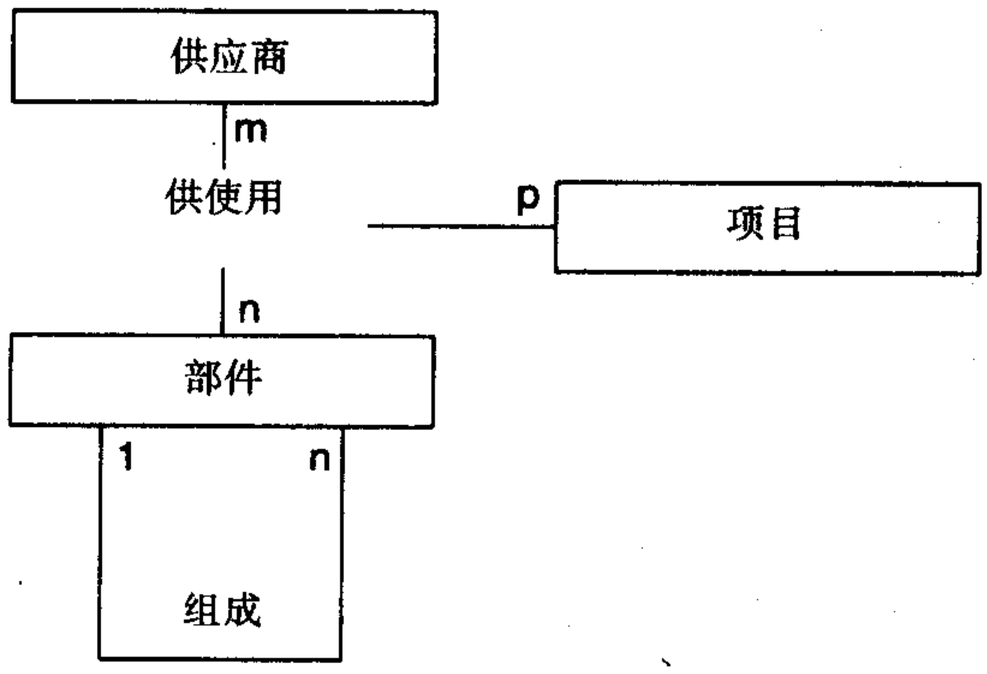

图8-10 更复杂的实体关系图

实体关系模型在下一章中进一步讨论。这一章将描述面向对象的分析，即另一种半形式化技术。本章的下一个主题是形式化技术。接下来的4节的基本主题是，使用形式化技术进行规格说明比半形式化和非形式化技术更精确。但是，形式化技术的使用一般需要长时间的训练，使用形式化技术的软件工程师需要相关的数学知识。此外，只要可能，应尽可能先使用数学公式，而不使用非正式的表示法。不过，8.6至8.9节的水平高于书中的其他部分。

8.6 有穷状态机

考虑下面例子，这个例子最初是由英国开放大学计算机系统的M202小组陈述的[Brady,1977]。一个保险箱有一个复合锁，有三个位置，分别为标为1、2、3，转盘能被向左或向右转动(L或R)。这样，在任意时刻，转盘都有6个可能的运动，即1L、1R、2L、2R、3L和3R。保险箱的组合密码是1L、3R、2L；任何其它的动作都将引起报警声。情况描述如图8-11所示。有一个初始态，即保险箱锁定(Safe Locked)状态。如果输入是1L，则下一个状态为A。但是，如果下一个转盘动作是1R或3L，则下一个状态为报警，这是两个终态之一。如果选择了正确的组合，则转换的序列为保险箱锁定到A到B，最后保险箱解锁(Safe Unlocked)，这是另外一个终态。图8-11是一个有穷状态机(FSM，Finite State Machines)的状态转换图(STD，state transition diagram)。状态转换图不一定要以图形方式描述，表8-1所示的表格形式化也可表达同样的信息。除两个终态以外的其它状态将根据转盘的转动方式转换到下一个状态。

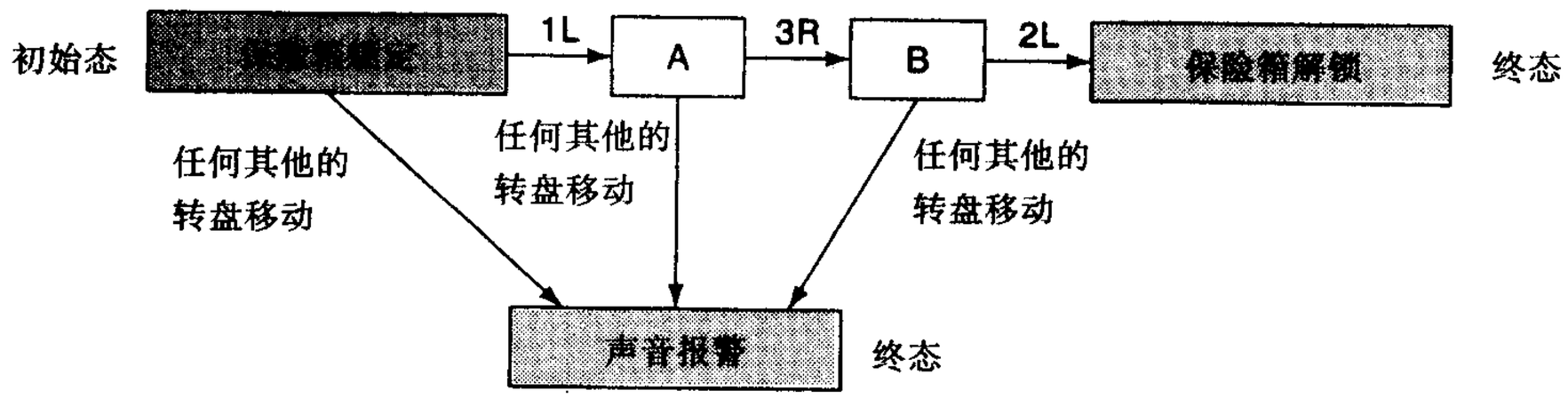

图8-11 组合密码保险箱的状态转换图

表8-1 有穷状态机转换表

当前状态 / 转盘动作	次态表 保险箱锁定	A	B
1L	A	声音报警	声音报警
1R	声音报警	声音报警	声音报警

（续）

当前状态 / 转盘动作	次态表 保险箱锁定	A	B
2L	声音报警	声音报警	保险箱打开
2R	声音报警	声音报警	声音报警
3L	声音报警	声音报警	声音报警
3R	声音报警	B	声音报警

一个有穷状态机包括5个部分：状态集J、输入集K、由当前状态和当前输入确定下一个状态(次态)的转换函数T、初始态S、终态集F。对于保险箱的组合锁来说情况如下：

状态集J：{保险箱锁定，A,B，保险箱打开，声音报警}

输入集：　{1L,1R,2L,2R,3L,3R}

转换函数T：如表8-1中的表格所示。

初始状态：　保险箱锁定

终态集F：　{保险箱打开，声音报警}

如果使用更形式化的术语，一个有穷状态机可表示为一个5元组(J,K,T,S,F)，其中：

J是一个有穷的非空状态集。

K是一个有穷的非空输入集。

T是一个从(J~F) * K到J的转换函数。

S∈J是初始状态。

F是终态集，F⊆J。

有穷状态机的使用在计算机化应用领域广泛传播。例如，每个菜单驱动的用户界面都是一个有穷状态机的实现。一个菜单的显示和一个状态相对应，通过键盘输入或用鼠标选择一个图标是一个使产品进入其他状态的一个事件。例如，当主菜单出现在屏幕上时，键入V可能引起对当前数据集进行定量分析。然后一个新菜单出现了，用户可以键入G、P或R，选择G将使计算结果图形化，P打印输出，R返回主菜单，每个转换都具有形式化：

当前状态[菜单] + 事件[所选择的项] => 下个状态　　(8.1)

为对一个产品进行规格说明，通常需要对有穷状态机进行一个很有用的扩展，即在前面的5元组中加入第6个组件：谓词集P，每个谓词都是产品的全局状态Y的函数。更形式化，转换函数T现在是一个从(j~F) × K × P到J的函数。现在的转换规则形式化如下：

当前状态[菜单] + 事件[所选择的项] + 谓词 => 下个状态　　(8.2)

为弄明白形式化工作在实际中是如何进行的，现在我们将把这一技术运用于电梯问题的修订版。可以看“8-I你想知道的进一步的信息”的内容，这些内容是关于这个问题的背景信息。

8-I　你想知道的进一步的信息

电梯问题是软件工程中的典型问题。它最早于1968年出现在Don Knuth的划时代著作《计算机程序设计技术——第1卷：基本算法》中的280～295页[Knuth,1968]。它是基于加利福尼亚技术学院数学系建筑中的一部电梯提出的。该例子用来解释在虚构的语言MIX中的协同程序。

到20世纪80年代中期，电梯问题被泛化为n部电梯，而且解决方案的特别属性必须要经过证明，例如一个电梯应该最终在有限的时间内到达。现在，任何形式化规格说明语言在提出时都不得不解决电梯问题。

该问题在1986年得到很大的发展，当时它发表在ACM SIGSOFT软件工程手册上，

是第4次关于软件规格说明和设计国际会议的论文[IWSSD,1986]。电梯问题是研究者引例的五个问题之一，这些例子被提交给1987年5月在加利福尼亚的蒙得里的亚举行的会议。在征集的论文中把它称为“提升”问题，并把它归功于STC-IDEC(在英国Stevenage的标准电信和电缆的一个部门)的N.Davis。

从那时起，这个问题又进一步获得广泛的重视，并被用来从整体上论证软件工程里大量的技术问题，而不仅仅是形式化规格说明语言。本书中用这个例子来解释各种技术，是因为读者不久就会发现，此问题决不像看起来那样简单。

电梯问题：有穷状态机

这个问题涉及到n台电梯根据下列约束条件在楼层间移动所需的逻辑：

C_1：每个电梯有m个按钮，一层一个。当这些按钮被按下时会发亮，使电梯访问相应的楼层。当电梯访问相应的楼层时，按钮就变暗关掉。

C_2：除了第一层和顶层之外，每一层都有两个按钮。一个请求向上，一个请求向下。这些按钮被按下时发光。当电梯到达该层时，发光的按钮立即熄灭；而后，电梯向要去的方向运动。

C_3：当电梯没有请求时，它停在当前层，同时让门关闭。

现在通过使用一个扩展的有穷状态机对本产品进行规格说明[Kampen,1987]。本问题里有两个按钮集。n个电梯中的每个电梯都有m个按钮，一个按钮对应一个楼层。因为这m×n个按钮都在电梯里，所以称它们为电梯按钮。另外，每层有两个按钮，一个请求向上，一个请求向下。这些按钮称为楼层按钮。

电梯按钮的状态转换图(STD)如图8-12所示。令EB(e，f)表示按下电梯e的按钮，并请求到f层去。EB(e,f)有两个状态，即按钮发光(打开)和不发光(关闭)。更精确地说，状态是：

EBON(e,f)： 电梯按钮(e,f) 打开 (8.3)

EBOFF(e,f)： 电梯按钮(e,f)关闭

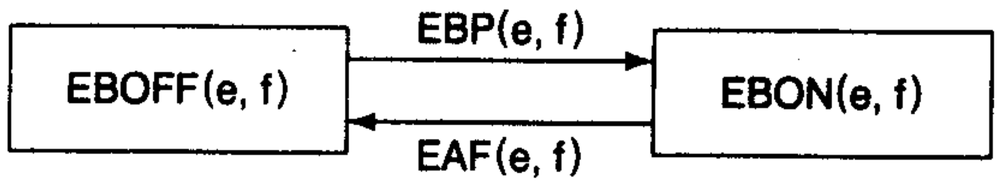

图8-12 电梯按钮的状态转换图[Kampen,1987]

如果按钮发光并且到达f层，按钮就会熄灭。相反，如果按钮熄灭，当按下它时，按钮又发光，这样就包含了两个事件，即：

EBP(e,f)：电梯按钮(e, f)被按下 (8.4)

EAF(e,f)：电梯e到达f层

为了定义与这些事件和状态相联系的状态转换规则，需要一个谓词V(e,f)(谓词是真或假的条件)。

V(e,f)：电梯e停在f层 (8.5)

现在可以描述形式化转换规则。如果电梯按钮(e,f)是关状态[当前状态]，并且电梯按钮(e,f)被按下[事件]，而且电梯不在访问f层[谓词]。那么，按钮打开发光，转换规则的形式化描述可以是：

EBOFF(e,f) + EBP(e,f) + not V(e,f) =>EBON(e,f) (8.6)

如果电梯当前正向f层移动，则什么也不会发生。在Kampen的形式化定义中，不触发状态转换的事件可能会发生；但是，如果确实发生了，也将被忽略。

反过来，如果电梯到达f层，而按钮又是打开的，于是它就会熄灭。这可做如下表示：

EBON(e,f) + EAF(e,f) =>EBOFF(e,f)　　(8.7)

现在考虑楼层按钮。FB(d,f)表示f层的按钮请求电梯向方向d运动。对于楼层按钮FB(d,f)的状态转换图如图8-13所示。

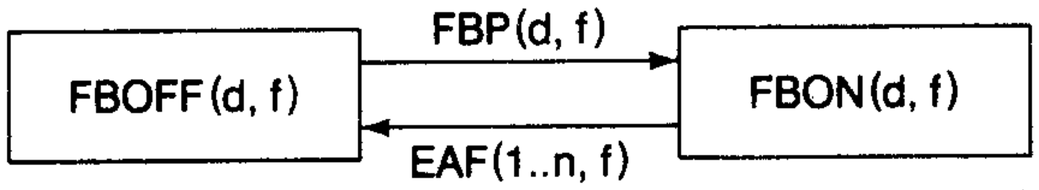

图8-13　楼层按钮的状态转换图[Kampen,1987]

确切地说，状态是：

FBON(d,f):楼层按钮(d, f) 打开　　(8.8)

FBOFF(d,f):楼层按钮(d, f) 关闭

如果按钮打开并且一个电梯以方向d到达f层，则按钮关闭。相反，如果按钮关闭并被按下，于是按钮被打开。同样，这也包含了两个事件，即：

FBP(d,f):楼层按钮(d, f) 被按下　　(8.9)

EAF(1..n,f):电梯1或…或n到达f 层

注意：1...n表示析取。在本节中，用表达式P(a,1..n,b)表示

P(a,1,b,) 或 P(a,2,b) 或… P(a,n,b)　　(8.10)

为定义与这些事件和状态相联系的状态转换规则，同样也需要一个谓词。在这种情况，它是S(d,e,f)，其定义如下：

S(d,e,f):电梯e停在f层并且移动方向由d确定为向上(d=U)、向下(d=D)或者待定(d=N)。　　(8.11)

这个谓词实际上是一个状态。事实上，形式化定义法允许将事件和状态作为谓词对待。

使用S(d,e,f)，则形式化转换规则为：

FBON(d,f) + FBP(d,f) + not S(d,1..n,f) =>FBON(d,f)　　(8.12)

FBON(d,f) + EAF(1..n,f) + S(d,1..n,f) => FBOFF(d,f),d=U or D.

也就是说，在f层朝方向d运动的楼层按钮是关闭状态，然后按钮被按下，并且没有正停在f层的电梯，这时运动方向为d，于是楼层按钮打开。相反，如果按钮打开，且至少有一个电梯到达f层，电梯将朝方向d运动，于是按钮关闭。标记1..n在S(d,1..n,f)和EAF(1..n.f)都有式(8.10)定义的含义。式(8.5)定义的谓词根据S(d,e,f)可按如下定义：

V(e,f) = S(U,e,f) or S(D,e,f) or S(N,e,f)　　(8.13)

定义电梯按钮和楼层按钮的状态都是很直观的。现在转向电梯，就会出现一些复杂情况。一个电梯状态本质上包括许多子状态。Kampen标识了几个，例如电梯减慢、停止、门打开、带延时的开门或在一段时间后自动关门[Kampen,1987]。他做了一些合理的假设：如果电梯控制器(决定电梯运动方向的机械装置)初始化为S(d,e,f)，然后控制器通过子状态移动电梯。现在定义了三个电梯状态，其中之一S(d,e,f)在式(8.11)中定义，但这里的定义更完备：

M(d,e,f):电梯e正沿d方向移动(f层是下一层)。　　(8.14)

S(d,e,f):电梯e停在f层

W(e,f):电梯e在f层等待(关了门)

三个状态如图8-14所示。注意，三个停止状态S(U,e,f)、S(N,e,f)和S(D,e,f) 已被组合成一个大的状态，目的是为了简化流图和减少状态总数。

可触发状态发生改变的事件为：DC(e,f),即电梯e在f楼层关门；ST(e,f)，即当电梯靠近楼层

f时，触发电梯上的传感器，电梯控制器决定在当前层电梯是否停止；RL，即电梯内按钮或楼层按钮被按下时进入ON状态。

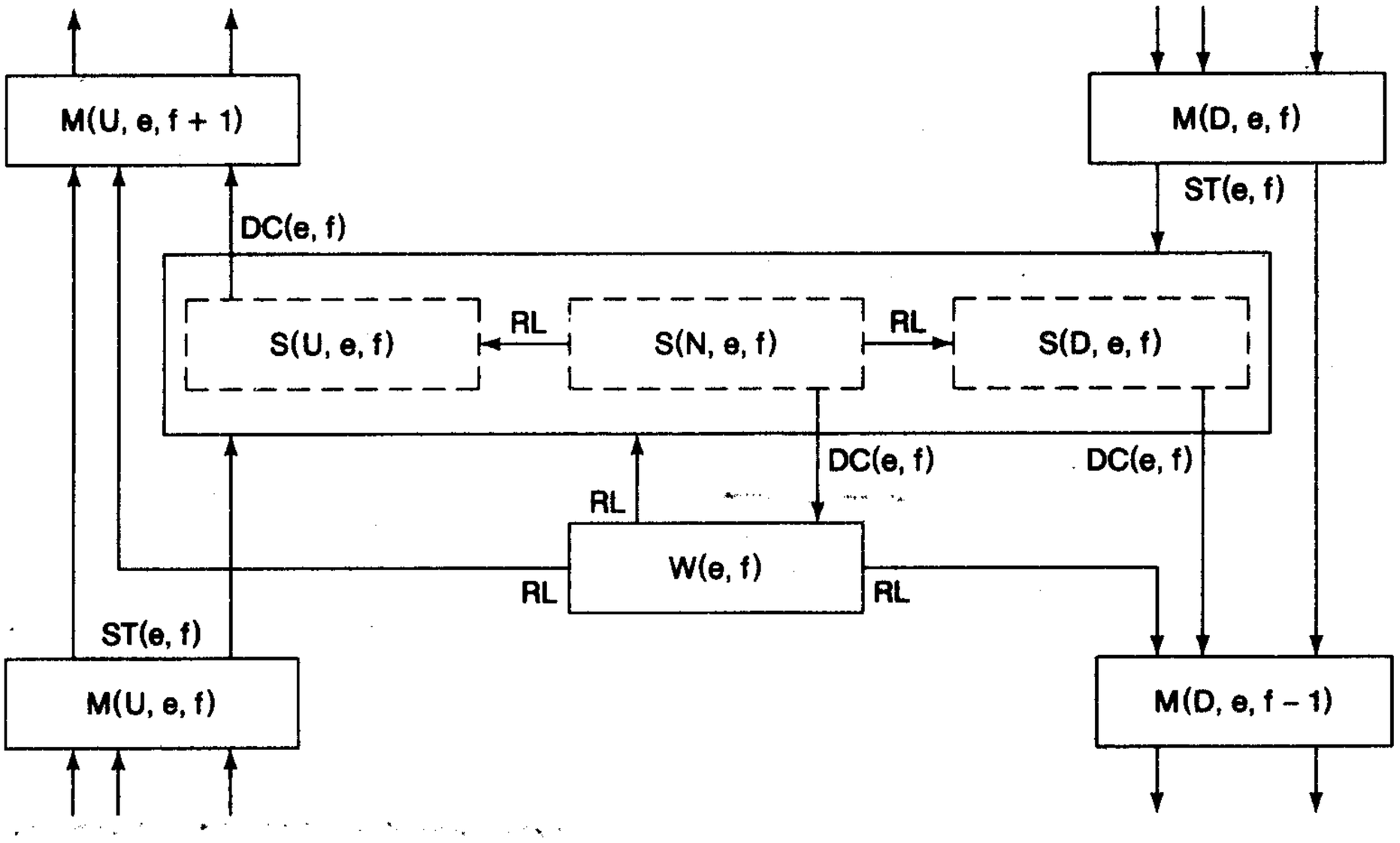

图8-14 电梯的状态转换图[Kampen,1987]

DC(e,f)：电梯e在楼层f关上门。 (8.15)

ST(e,f)：电梯e靠近f层时，传感器被触发。

RL：登录需求(按钮被按下时)。

这些事件都表示在图8-14中。

最后，给出电梯的状态转换规则。这些规则可以由图8-14推出，不过在有些情况下，需要做出额外的谓词。如果要求更精确，则图8-14就显得不太精确了。在某些推理中，必须利用谓词来使状态转换图更具确定性。有兴趣的读者可以参见[Kampen,1987]，以了解完整的规则。为简洁起见，这里所提出的规则只是发生在关门之时。电梯的上升、下降或进入等待状态，都取决于当前的状态。

S(U,e,f) + DC(e,f) =>M(U,e,f+1) (8.16)

S(D,e,f) + DC(e,f) =>M(D,e,f-1)

S(N,e,f) + DC(e,f) =>W(e,f)

第一条规则表明，如果电梯e处于状态S(U,e,f)，即停止在f层，门已经关闭，并且准备上升，则电梯将往上一楼层移动。第二条与第三条规则对应于电梯即将下降或者没有要处理的请求的情况。

这些规则的格式反映了有穷状态机在说明复杂产品方面的能力。规格说明并非是列出一系列复杂的前置条件让产品做什么，然后列出产品处理后的全部状态；而是采用了一种简单的描述格式：

当前状态 + 事件+ 谓词=>下个状态

这种类型的规格说明易于书写、易于验证并可容易地转换为设计或代码。其实，可以直接构造一个CASE工具将一个有穷状态机规格说明直接转化为源代码。维护可以通过重现来实现。也就是说，如果需要一个新的状态或事件，规格说明将被修改，则新版本产品就直接由新的规

格说明产生。

FSM(有穷状态机)方法比8.3节中列出的Gane和Sarsen技术更精确，并且和它一样易于理解。不过，它有一个缺点，就是在一个大系统中三元组(即状态、事件、谓词)的数量会迅速增长。此外，与Gane和Sarsen的方法一样，Kampen的形式化方法对定时需求也未加以处理。

这些问题都可通过使用FSM的一个扩充[Harel et al.,1990]——状态图来解决。状态图的功能很强，而且为CASE工具平台及状态平台所支持。这种方法已成功地用于一些大型的实时系统中。

另一种可处理定时问题的形式化方法是Petri网。

8.7　Petri网

并发系统中一个主要问题就是定时问题。这一问题可以表现为多种形式化，如同步问题、竞争条件以及死锁[Silberschatz and Galvin,1994]。定时问题经常是由不好的设计或有错误的实现引起的，而这些设计与实现通常又是由不好的规格说明引起的。如果规格说明制订得不恰当，则将产生相应的设计或实现不完备的危险。用于确定系统中隐含的定时问题一种有效的方法是Petri网。这种技术一个很大的优点是它也可以用于设计中。

Petri网是由Carl Adam Petri[Petri,1962]发明的。一开始只有自动化专家对Petri网感兴趣，后来，Petri网在计算机科学中得到了广泛的应用，如在性能评价、操作系统和软件工程等领域内的应用。特别是Petri网被证明可有效地描述并发关系活动。但是，在使用Petri网做规格说明前，需要向那些不太熟悉这种方法的读者做一个简单的介绍。

Petri网包含4个部分：一组位置P、一组转换T、输入函数I以及输出函数O，如图8-15所示。

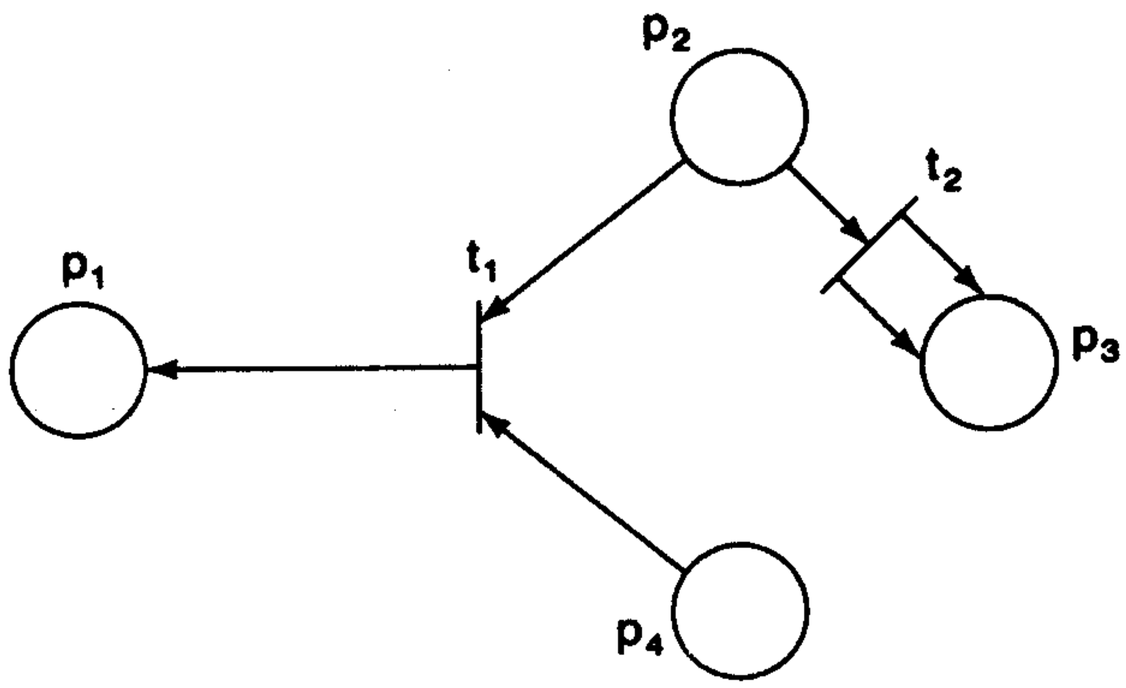

图8-15　Petri网

一组位置P为$\{p_1,p_2,p_3,p_4\}$

一组转换T为$\{t_1,t_2\}$

两个用于转换的输入函数由位置指向转换的箭头表示，它们是：

$I(t_1)=\{p_2,p_4\}$

$I(t_2)=\{p_2\}$

两个用于转换的输出函数由转换指向位置的箭头表示，它们是：

$O(t_1)=\{p_1\}$

$O(t_2)=\{p_3,p_3\}$

注意，有两个p_3，是因为有两个箭头由t_2指向p_3。

更为形式化的Petri网结构[Peterson,1981]为一个四元组C=(P,T,I,O)。

$P=\{p_1,...,p_n\}$是一个有穷位置集，$n \geqslant 0$。

$T=\{t_1,t_2,...,t_m\}$是一个有穷转换集，$m \geqslant 0$，且P和T不相交。

$I:T \rightarrow P^\infty$为输入函数，是由转换指向位置无序单位组的(bags)映射。

$O:T \rightarrow P^\infty$为输出函数，是由转换指向位置无序单位组的映射(一个无序单位组或多重组是允许一个元素有多个实例的广义集)。

Petri网的标记是指在Petri网中令牌(token)的分配。在图8-16中有4个令牌，一个在p_1中，两个在p_2中，p_3中没有，还有一个在p_4中。标记可用向量(1,2,0,1)表示。p_2与p_4中有令牌，则t_1启动(即被激发)。通常情况下，当每个输入位置所拥有的令牌等于位置到转换的线的数量时，就允许转换。当t_1被激发时，p_2与p_4上各有一个令牌被移出，而p_1上则增加一个令牌。令牌的数目不是保留的，两个令牌被移出，而p_1上只能新放入一个令牌。图8-16中p_2上有令牌，则t_2也可被激发。当t_2被激发时，p_2上将移走一个令牌，而p_3上则新增加两个令牌。

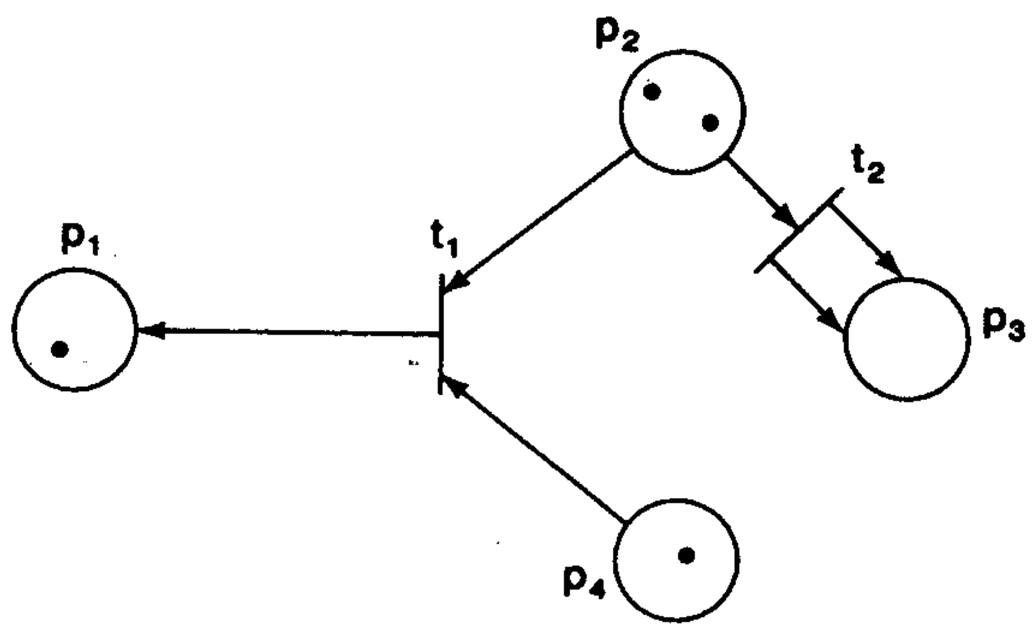

图8-16 带标记的Petri网

Petri网是非确定性的，即数个转换都达到激发条件，则其中任意一个都可以被激发。图8-16的标记为(1,2,0,1)，t_1和t_2都可被激发，假设t_1被激发了，则结果为图8-17所示，标记为(2,1,0,0)。此时，只有t_2可被激发。如t_2也被激发，则启动令牌从p_2中移出，而两个新令牌被放置在p_3上。如图8-18所示。标记为(2,0,2,0)。

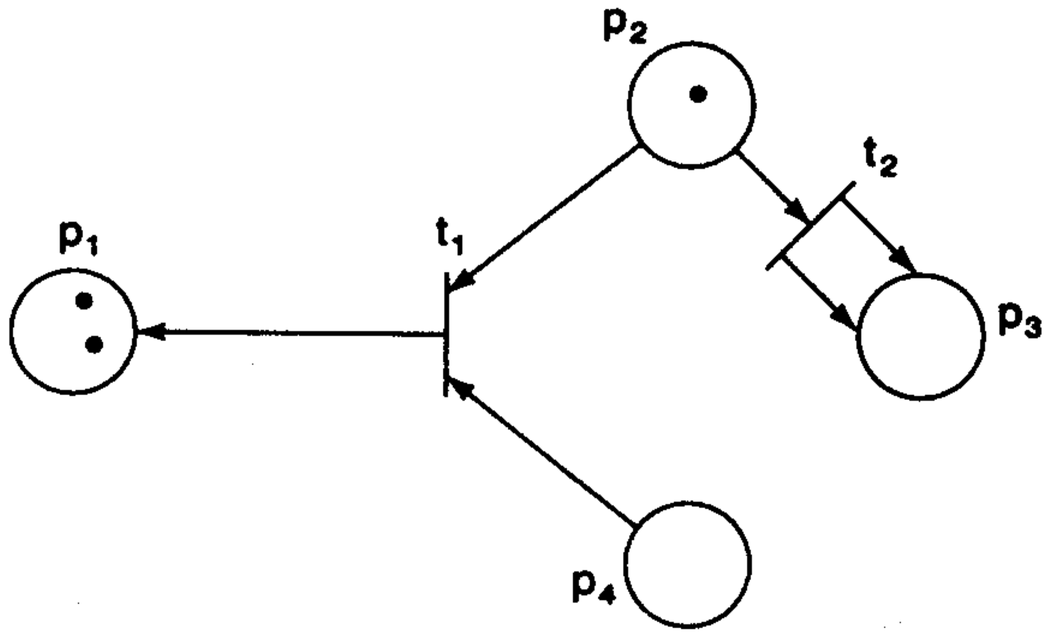

图8-17 图8-16的Petri网在转换t_1激发后的情况

更形式化地[Peterson,1981]，Petri网C=(P,T,I,O)中的标记M(marking)是由一组位置P到一组非负整数的函数，

$M:P \rightarrow \{0,1,2,...\}$

这样，带有标记的Petri网成为一个五元组(P,T,I,O,M)。

Petri网的一个重要的扩充是加入禁止线。如图8-19所示，禁止弧是用一个小圆圈而不是箭头标记的。因为p_3上有一个令牌而p_2上没有令牌，所以转换t_1是允许的。通常情况下，每个输入线上至少有一个令牌，而禁止线上无令牌，则一个转换才是允许的。Petri网的这一扩展将在下面关于电梯问题的规格说明中进行介绍[Guha,Lang,and Bassiouni,1987]。

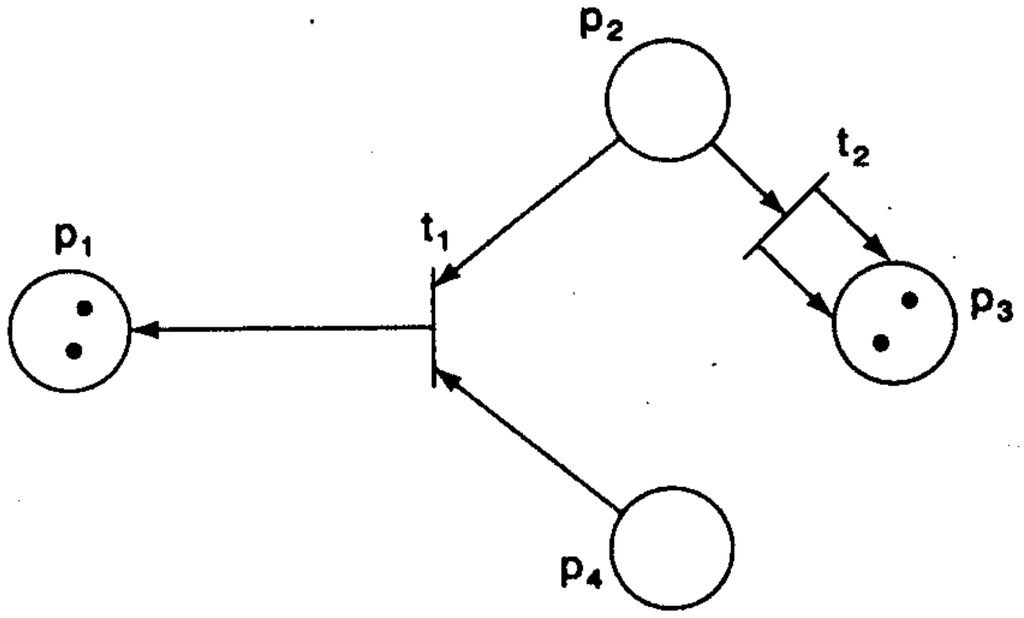

图8-18　图8-17中的Petri网在转换t_2激发后的情况

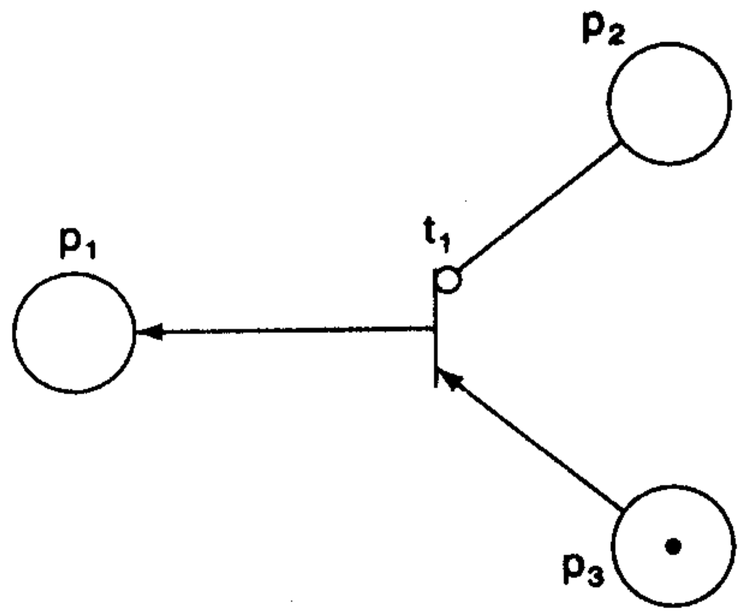

图8-19　含禁止线的Petri网

电梯问题：Petri网

回想一下在一座m层的建筑中装了n个电梯系统的例子。在Petri网规格说明中，每一层用一位置F_f代表(1<f<m)，在Petri网中，电梯是用一令牌代表的。在F_f上有令牌，则表示在楼层f上有电梯。

△　第一条限制：每个电梯具有m个按钮，每层对应一个。当按下按钮时，指示灯会发亮，指示电梯将移往相应的楼层。而当电梯到达相应的楼层时，按钮将熄灭。

为了将其与规格说明相协调，还必需设置其他的位置。电梯中的楼层f的按钮在Petri网中是用EB_f表示的($1 \leqslant f \leqslant m$)。此外，由于有n套电梯，则位置就表示为$EB_{f,e}$($1 \leqslant f \leqslant m$，$1 \leqslant e \leqslant n$)。但为了描述简洁些，用于表示电梯的下脚标e将省去。在EB_f上有一令牌，则表示电梯内楼层f的按钮被按下。因为按钮只有在第一次被按下时才亮，而后来再按也只会被忽略。这些已在图8-20 Petri网的使用中详细说明了。首先，假设按钮没有发亮，显然位置上无令牌，从而在存在禁止线的情况下，转换“EB_f被按下”是允许的。如图8-20所示，现在按下按钮，则转换被激发，并在EB_f上放置了一个新的令牌。现在不论按钮再被按下多少次，禁止弧与现有令牌间的组合都决定了转换“EB_f被按下”不能再被允许了。从而位置EB_f的令牌数不会多于1。假定电梯由g层驶向f层，因为电梯在g层，如图8-20所示，位置F_g上有一令牌。转换“电梯在运行”

(Elevator inaction)被激发，EB_f与F_g上的令牌被移去，从而按钮EB_f被关闭，在F_f上发现一个新令牌，转换的激发使电梯由g层驶至f层。

由g层移到f层并不能立即发生。为处理这个情况以及类似的问题，例如，因为物理上的原因使按钮不能按下后马上发亮，Petri网模型中必须加入时限。这就是说，在标准Petri网中转换是瞬时的，而在现实情况下，例如，电梯问题就需要时间控制Petri网[Coolahan and Roussopoulos,1983]，以使转换与非零时间相结合。

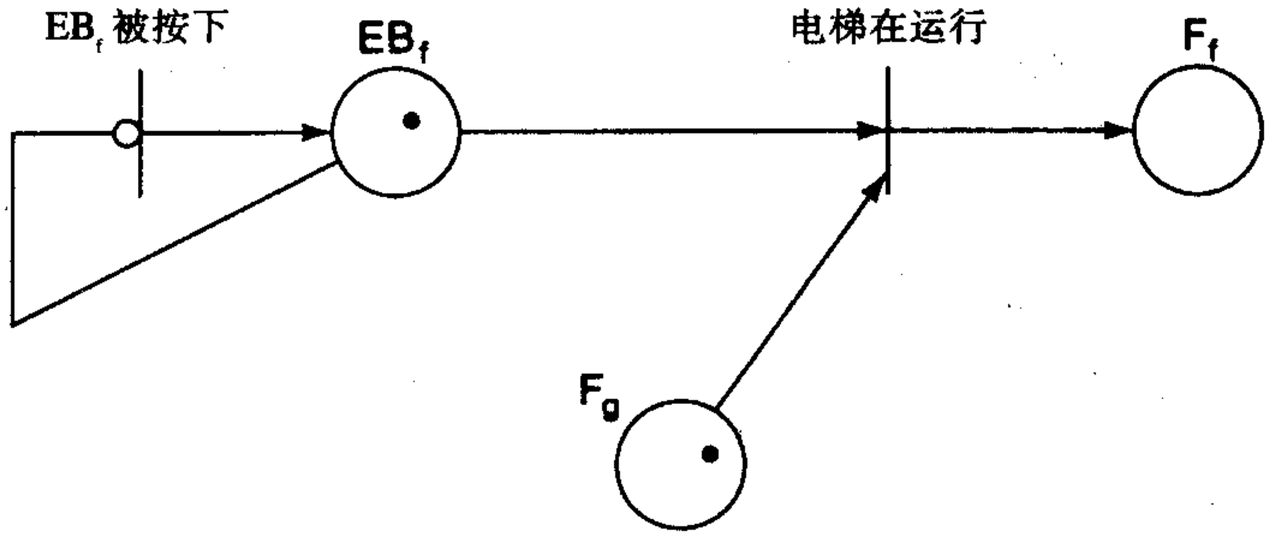

图8-20 Petri网表示的电梯按钮

△ 第二条限制：除了第一楼层与顶层外，每个楼层都有两个按钮，一个要求电梯上行，另一个要求电梯下行。这些按钮在按下时发亮，当电梯到达该层并移向指定方向时，灯才会熄灭。

楼层按钮用位置FB^u_f及FB^d_f表示，分别代表上行与下行的电梯。此外，底层的按钮为FB^u_1，最高层m的按钮为FB^d_m，中间每一层有两个按钮FB^u_f及FB^d_f(1<f<m)。如图8-21所示，情况为电梯由g层到达f层，有一个或两个按钮都亮。如果两个按钮都亮了，则只有一个按钮熄灭。要保证按钮熄灭正确，则需要更复杂的Petri网模型；参看[Ghezzi and Mandrioli,1987]。

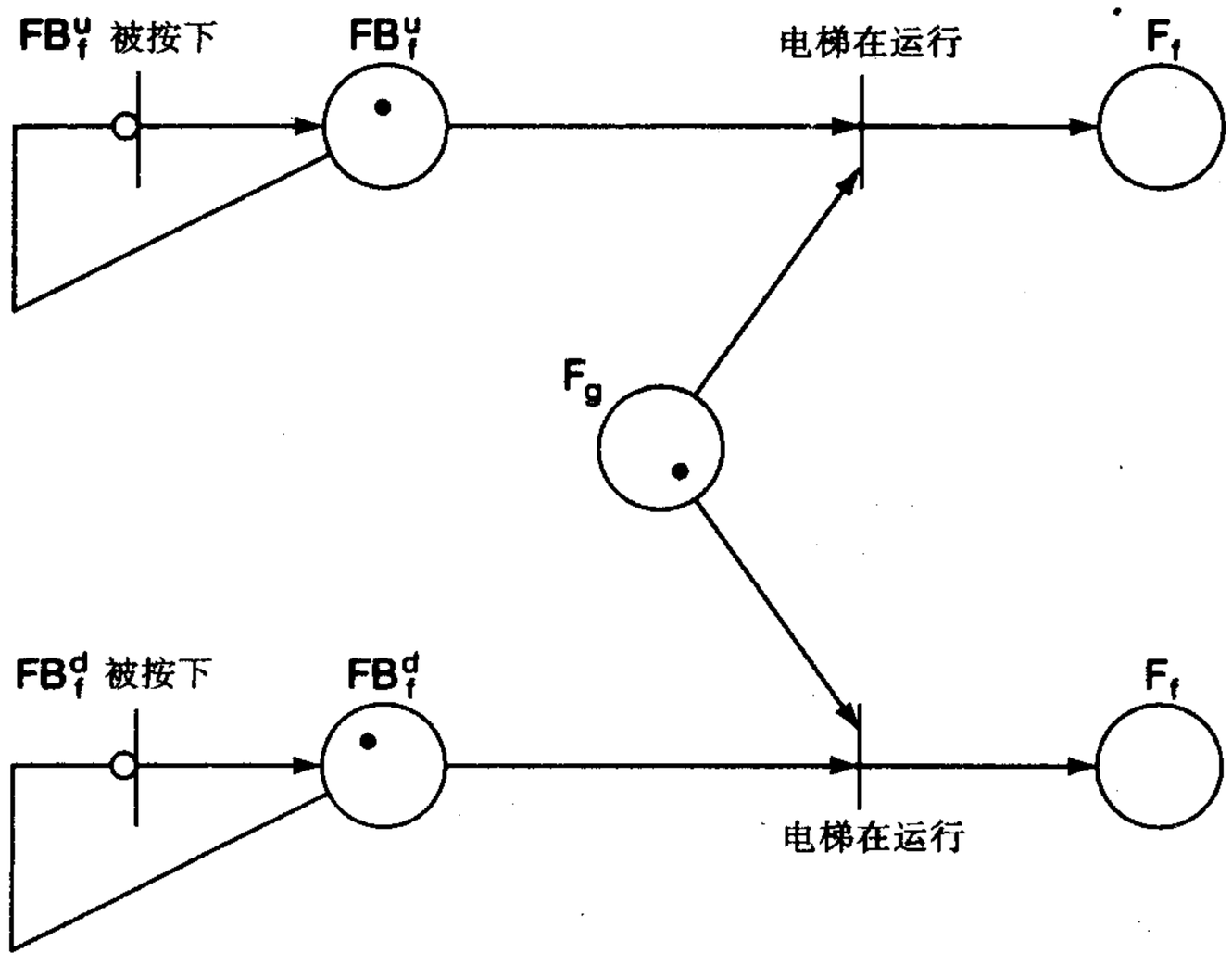

图8-21 Petri网表示楼层按钮 [Guha Lang and Bassioini,1987]

△ 第三条限制：当电梯没有收到请求时，它将停留在当前楼层并关门。

这条限制很容易实现。当没有请求时，没有转换“电梯在运行”被允许。

Petri网不仅可以用于规格说明，同样可以用于设计[Guha Lang and Bassiouni,1987]。即使在产品的开发阶段，也可以清楚地知道Petri 网拥有用于确定并发系统同步性能的能力。

8.8 Z

在形式化规格说明语言中，赢得广泛声誉的是Z[Spivey,1992](要了解名称Z的发音请参看“8-II 你想知道的进一步的信息”)。使用Z需要了解集合论、函数、离散数学，包括第一级逻辑的知识。即使用户已掌握了所需要的背景知识(包括许多计算机专业人员)，开始时Z也是很难学的。因为它除了含有常用的集合论和逻辑符号，例如∃、⊃或=>，还使用一些特殊符号，例如⊕、⩤、↦ 和 ↠ 。

为了了解Z是如何用来说明一个产品的，需要再一次对8.6.1节中的电梯问题进行讨论。

8-II 你想知道的进一步的信息

Z是其发明者Jean-Raymond Abrial给形式化规格说明语言命名的名称，其目的是为了记念著名的集合论专家Ernst Friedrich Ferdinand Zermelo(1871-1953)。因为它是在牛津大学开发的[Abrial,1980]，采用了字母表的第26个字母的英式发音，所以名称Z的准确发音为“zed”。

后来，有人证实Z是以一位德国数学家的名字命名的，用德文方式发音则为“tzed”。与此对应，Francophiles和Francophones指出Abrial是一个法国人，则字母Z应按法式发音，也为“zed”。

美式发音“zee”没有被人们所接受，因为Z(发音“zee”)是一种美国第四代语言(见12.2节)。更准确的是，System Z是第四代语言Z的工作平台，System Z是Zortec国际公司的商标。然而我们不能以字母表中的一个字母为商标。此外，我们可以以任何喜欢的方式读字母Z。尽管如此，在编程语言环境中，发音“zee”指的是这种第四代语言，而并非形式化规格说明语言。

8.8.1 电梯问题：Z

最简单的Z规格说明形式化包含4个部分，即：

1) 给定的集合、数据类型及常数。
2) 状态定义。
3) 初始状态。
4) 操作。

现在对这4部分依次加以分析。

1) 给定的集合：一个Z规格说明从一系列给定的初始化集合开始，即不需要详细定义的集合。这种集合以带方括号的形式表示。对电梯问题，给定的初始化集合称为Button，即所有按钮的集合。因此，Z规格说明开始于：

[Button]

2) 状态定义：一个Z规格说明包含多个格(schema)，每个格又含有一组变量说明和一系列限定变量的取值范围的谓词。格s的格式如图8-22所示。

在电梯问题中，Button有四个子集，即floor_buttons、elerator_buttons、buttons (电梯问题中所有按钮的集合)以及pushed(所有被按的按钮的集合，或者说这些按钮是处于打开状态)。

图8-23描述了格Button_State。符号P表示幂集(给定集的所有子集)。约束条件声明：floor_buttons与elevator_buttons集不相交，且它们共同组成buttons 集(下面将不需要floor_buttons集与elevator_buttons集，将它们包含在图8-23中只是用来表示Z的幂)。

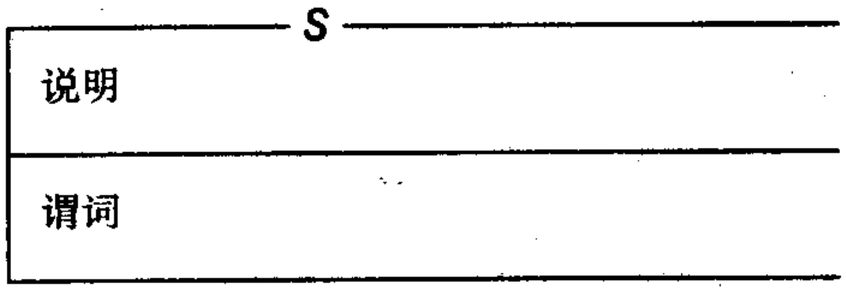

图8-22 Z格s的格式

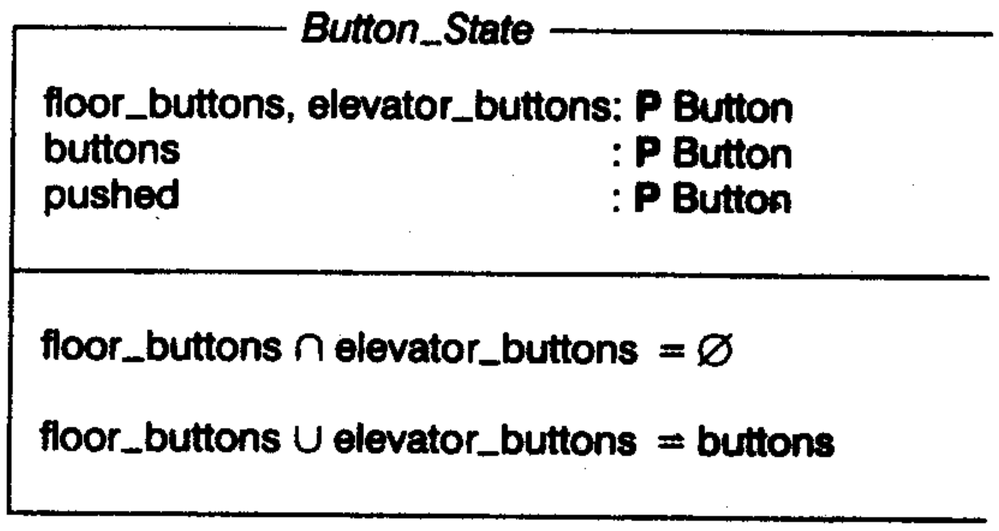

图8-23 Z格Button_State

3) 初始状态：抽象的初始状态是指系统第一次开启时的状态。电梯问题中抽象的初始状态为：

$$Button\_Init \triangleq [Button\_State' \mid pushed' = \emptyset]$$

如图8-23所示，这是垂直格的定义，与水平格定义相对应。垂直格定义是指当系统首次开启时，pushed集为空，即所有按钮都关闭。

4) 操作：如果一个按钮首次被按下，则按钮开启。这个按钮就被添加到pushed集中；这一过程如图8-24所示。图中定义了操作Push_Button。格的第一行中的三角表示该操作改变了Button_State的状态。该操作有一输入变量，即Button？。与其他语言(例如CSP)一样，问号？表示输入变量，而感叹号！则代表输出变量。

操作的谓词部分包含了一组操作被调用的前置条件，以及操作完全结束后的后置条件。若前置条件成立，则执行完成后可得后置条件。然而，如果在前置条件不满足的情况下就调用该操作，则无法得到指定的结果(因此结果无法预测)。

图8-24中的第一条前置条件规定button?必须是buttons的一个元素，而buttons是指电梯系统中所有按钮的集合。如果第二条前置条件Button?∉pushed得以满足(即按钮没有开启)，则更新pushed按钮集，使之包含Button?。在Z中，当一个变量的值发生改变时，就用符号“'”表示。因此，后置条件是：当执行完操作Push_Button之后，button?将被加入pushed集。这时无需直接打开开关，只要button?变为pushed中的一个元素即可。

另一种可能性是按钮已被按了一下。由于button? ∈ pushed，根据第三条前置条件[Θ]，将没有任何事发生。这可用pushed'=pushed表示，即pushed的新状态与旧状态一样。

假设电梯到达了某层楼，如果相应的楼层按钮已打开，则此时它会关闭，而相应的电梯按钮也是如此。如果button?属于pushed集，则它将被移出集合，这些可参见图8-25(符号\表示集

Θ 表如果没有第三个前置条件，规格说明将不能说明在一个按钮已被按下后又被按了一次的情况下将发生什么。这样，结果将是不可预测的。

合相异)。然而，如果按钮没有打开，则pushed不发生变化。

Push_Button

ΔButton_State

button?: Button

(button? ∈ buttons) ∧
(((button? ∉ pushed) ∧ (pushed′ = pushed ∪ {button?}))∨
((button? ∈ pushed) ∧ (pushed′ = pushed)))

图8-24　操作Push_Button 的Z规格说明

Floor_Arrival

ΔButton_State

button?: Button

(button? ∈ buttons) ∧
(((button? ∈ pushed) ∧ (pushed′ = pushed \ {button?}))∨
((button? ∉ pushed) ∧ (pushed′ = pushed)))

图8-25　操作Floor_Arrival 的Z规格说明

本节的解答有所简化，它不能辨别上行和下行楼层按钮，然而它演示了如何使用Z来说明电梯问题中按钮的状态。

8.8.2　对Z的分析

Z已在很多项目中成功地运用，包括CASE工具[Hall,1990]、实时内核[Spivey,1990]以及示波器[Delisle and Garlan,1990]。在新版CICS(IBM 事务处理系统)中，有很大一部分是使用Z进行规格说明的[Nix and Collins,1988]。

虽然有了这些成功，但你也许会对现实情况感到惊讶，因为即使在电梯问题的简化版本中，Z也无法直接使用。首先是由符号引起的问题。一个新用户在阅读Z规格说明以前，必须了解符号集及其含义。其次是，并非每个软件工程人员都了解足够使用Z的数学知识(虽然近年来，计算机科学专业的毕业生或已经掌握了使用Z的足够数学知识，或可以比较容易地学会所需要的部分)。

Z也许是应用最广泛的形式化语言。这是为什么呢，Z为何获得如此多的成功，尤其是在大型项目中？原因有以下几点。

首先，用Z写的规格说明的错误可以比较容易地被发现，尤其是自己审查规格说明，及根据形式化规格说明来审查设计与代码 时[Nix and Collins,1988；Hall,1990]。

第二，用Z写规格说明时，要求规格说明人员要十分精确地使用Z说明符。由于对精确性要求很高，从而和非形式化规格说明相比减少了模糊性、矛盾和遗漏。

第三，Z是一种形式化语言，允许开发者在需要时验证规格说明的正确性。虽然有些公司很少做正确性验证，但这种验证实际上已经做了，即使是在如CICS存储管理器之类的实际系统中。

第四，经证明，那些只学过中学数学的软件专业人员，可以用相对短的时间学会编写Z规

格说明[Hall,1990]。很明显，这些人无法证明规格说明的结果是否正确。但是，形式化规格说明也并非总要做正确性证明。

第五，使用Z可以降低软件开发费用。虽然花在规格说明上的时间无疑要比使用非形式化技术要多，但开发过程中所需的总时间是降低的。

第六，有几种途径来解决用户无法理解用Z写的规格说明这一问题，其中包括用自然语言重写规格说明。人们发现，由此得到的自然语言规格说明比凑合写出的非形式化规格说明要清楚(这一点和8.2.1节中Meyer用英语来解释Naur的文本处理问题的形式化规格说明的经验一样，参见8.2.1节)。

最后，虽然有相反的观点，但是，Z已在软件工业的一些大型项目中成功地运用。尽管还有大量的规格说明是用没有Z那样规范的语言编写的，但使用形式化规格说明是全球的总趋势。使用形式化规格说明过去主要是欧洲的习惯，然而越来越多的美国公司也使用了一些形式化规格说明技术。Z及类似的语言在将来的使用程度尚须拭目以待。

8.9 其他的形式化技术

许多其他的形式化技术也已提出，这些技术之间差别非常大。如Anna[Luckham and von Henke,1985]是一种用于Ada的形式化规格说明语言。一些形式化技术是基于知识的，如Refine[Smith,Kotik,and Westfold,1985]以及Gist[Balzer,1985]。Gist的设计思路是使用户可以用近似于思考过程的方式来描述处理过程。这是通过对自然语言中所使用的结构进行形式化而获得的。实际上，Gist规格说明也和几乎其他所有的规格说明一样难读，从而将Gist解释为英语也很难写。

Vienna定义方法(VDM，Vienna definition method)[Jones,1986b;Bjorner,1987]是一种基于标志语义的技术[Gordon,1979]。VDM不仅可用于规格说明，而且同样可以用于设计和实现。VDM已在一些工程中成功地运用，其中最著名的是Dansk Datamatik Center开发的DDC Ada编译系统[Oest,1986]。

看待规格说明的另一种方法是根据事件发生的顺序，事件或者是简单的行为,或者是系统输入输出数据的通信。例如，在电梯问题中，一个事件是由在电梯e中按下要去f层的按钮，按钮发亮所组成的。另一个事件就是电梯e离开楼层f并下行，按钮熄灭。通信序列处理(CSP，Communicating Sequential Processes)语言是由Hoare发明的，其基本思想是按上述事件来描述系统状态[Hoare,1985]。在CSP中，处理是按照事件的顺序来描述的；在一些事件序列中处理将和环境交互。处理之间通过传输信息交互作用。CSP允许处理通过多种方式进行组合，如顺序地、并行地或非确定性地交错。

CSP的强大在于：CSP规格说明是可执行的[Delisle and Schwartz,1987]，从而可检查文档内部的一致性。此外，CSP提供了一个框架，通过一系列步骤保证从规格说明到设计再到实现这一过程的有效性。换句话说，如果规格说明是正确的，并且转换又是正确的，则设计与实现也将是正确的。如果实现语言是Ada，则设计到实现就可直接进行。

然而CSP也有其缺点。特别是与Z一样，它也不是一种容易学习的语言。在本书中尝试在电梯问题中使用CSP规格说明[Schwartz,Delisle,1987]，然而要对每个CSP命题都充分描述的话，所需要的基本预备材料量及细节说明都会太多，从而无法将其全部包含在本书中。一种规格说明的能力与使用它的难度之间的关系将在下一节展开讨论。

8.10 规格说明技术的比较

本章的主要内容是帮助各个开发公司了解何种规格说明语言适合于其所要开发的产品。非形式化技术很容易学会，但达不到半形式化的或形式化技术所具有的功能。相反地，每种形式化技术都支持各种不同的特性，包括可执行性、正确性证明，或者通过一系列保证正确性的步骤转换为设计及实现。虽然通常情况下技术越形式化，则功能越强，然而形式化技术通常也难以掌握和使用。同样，用户也很难理解一个形式化规格说明。换句话说，需要有一种规格说明的应用性与功能间的权衡。

在某种情况下，确定规格说明的类型很容易。例如，如果大部分开发者并未受过计算机科学方面的教育，则实际上只能使用非形式化的或半形式化的技术。相反的，在研究室或实验室中构建关键任务的实时系统时，则必须利用形式化规格说明的强大功能。

另外一个复杂的因素是，许多新的形式化技术并未经过实践条件的检验，使用这样的技术会有很大风险。为了培训开发组中的相关人员，需要花很多钱。而在开发队伍将这些在课堂上学会使用的语言应用到实际项目中时，将花费更大。例如，在SREM中[Scheffer,Stone,and Rzepka,1985]，由于所用语言的软件支持工具不能正常工作，结果造成额外的花费和时间上的延期。但是，如果各部分都能正常工作，而且在软件项目管理方案中,对一项新技术首次在重大项目中使用时所需的额外时间和经费加以考虑的话，则收益可能是巨大的。

在具体项目中应使用何种规格说明技术呢？这要取决于项目、开发组、管理人员以及其他许多的因素，例如，用户可能坚持要使用(或不使用)某种方法。和软件工程中的其他方面一样，必须在各种方案中做出折衷。不幸的是，决定使用何种规格说明技术并不存在一条简单的规则。

8.11 规格说明阶段的测试

在规格说明阶段，处于酝酿中的产品的功能要在规格说明文档中精确地表达出来。验证规格说明文档的正确性是很重要的，对规格文档进行走查是达到这一目标的方法之一(5.2.1节)。

检测规格说明文件中错误的一个更有效的机制是审查(5.2.3节)。一组审查员根据检验清单审阅规格说明。在规格说明的检验清单中典型的条目包括：是否规定了所需的硬件资源？是否确定了验收标准？

审查最早是由Fagan在他的关于检测技术与代码的文章中提出的[Fagan,1976]。Fagan的文章对审查做过详细描述。然而，审查已被证明在检测规格说明中是相当有效的。例如，Doolan利用审查对一个有两百万FORTRAN语句行的产品的规格说明进行检测[Doolan,1992]。从修改产品错误的数据中，他已可以减少审查过程中每小时的投资，并节省了30小时用于执行错误的测试与纠正的时间。

当一个规格说明采用形式化技术拟定后，也可使用其他的测试技术进行测试。例如，可使用正确性证明方法(5.5节)。即使不使用形式化证明方法，非形式化证明技术(如5.5.1节中所使用的技术)对发现规格说明错误也很有用处。实际上产品及其证明应并行开发。这样，错误就能很快地找出。

8.12 规格说明阶段的CASE工具

在规格说明阶段有两类CASE工具特别有用。第一类是图形工具。无论产品是使用数据流

程图、Petri网、实体关系图或本书因篇幅有限而未列出的其他表示法描述的，手工画出整个产品都将是一个冗长的过程。此外，在做一些重大的修改时，所有的部分都必须从头重画。因此，使用绘图工具将节约大量的时间。这些工具适用于本章描述的规格说明技术，还有其它一些图形表示法对规格说明很有用。在此阶段的另一类工具为数据字典。如4.3.1节所述，这种工具存储产品中各个数据项的每一成份的名称和表示法(格式)，包括：数据流程及其组成、数据存储及其组成、处理及其内部变量等。在各种硬件上有大量可选用的数据字典。

真正需要的并非是独立的图形工具或独立的数据字典。这两种工具应该集成起来，从而使数据成分的任何变化都将自动地反映到规格说明的相应部分。这里有许多此类工具的例子，较常见的有ADW、Analyst/Designer、Excelerator、Software through Pictures、System Architect及Teamwork。此外，许多类似的工具还与一致性自动检测工具相结合，从而保证规格说明文档与相应的设计文档间的一致性。例如，可以检验规格说明文档中的每一项是否已被带到设计文档中，且在设计中提到的所有东西都在数据字典中被说明过。

一项规格说明技术，除非得到了一个工具丰富的CASE环境的支持，否则无法被广泛接受。例如，如果与SREM(8.4节)相关的CASE工具集REVS能在美国空军测试中表现得更好的话[Scheffer,Stone, and Rzepka,1985]，SREM现在将使用得更为广泛。即使是有经验的软件专业人员也很难准确地对一个系统进行规格说明，只有给规格说明人员提供一套现代化的CASE工具，以在各方面帮助他，这样他才可能正确地进行规格说明。

8.13 规格说明阶段的度量

和其他阶段一样，在规格说明阶段也需度量五个基本的方面，即规模、成本、期限、工作量及质量。度量规格说明规模的一种方法是计算规格说明文档的页数。如果整个产品使用的技术相同，则可通过规格说明大小来有效地预测制造产品所需要的工作量。

对于质量，规格说明的一个至关重要的方面是故障统计记录。因此，在审查过程中发现的各类错误的数量是审查过程的一个组成部分。同时，错误的检测率也可度量审查过程的效率。

预测目标产品大小的尺度包括数据字典中的条目数。要考虑多种不同的量，包括文件数、数据条目数、处理量等。这些信息使管理人员可初步估算出在该产品上要付出的工作量。要注意，这些信息只是暂定的。毕竟，在设计阶段，数据流图中的一个处理可以分解成几个不同的模块。相反，几个处理也可能合在一起构成一个新的模块。尽管如此，从数据字典中得出的度量，可以给管理人员有关目标产品最终大小的早期线索。

8.14 MSG实例研究：结构化系统分析

对于需求分析阶段，Gane与Sarsen 的结构化分析过程很直观，就是经过一系列细致的会谈、分析和抽象后，可构造出一个快速原型。MSG数据流程图如图8-26所示。而结构化分析的其余部分见附录D。

规格说明的一个方面需要进一步加以说明。在数据流程图中，如图8-26所示，两个数据存储为收入数据和津贴数据。这两个数据存储在第5步骤中被求精，且其物理结构在第6步确定。其实，收入数据或津贴数据都无需作为一个文件来实现。收入数据包含两个量，即下一个星期期望的收入与期望的抵押支付。津贴数据为下一个星期预期可支用的津贴的数量。这三个量是通过处理“计算每周预望收入、抵押付款及津贴”来计算的。在理论上它们可以被当作全局变量存储，或作为参数由这个处理传递到处理“计算可用资金并产生资金报告”。然而这种设计

并不是一个好主意，这有两个原因。首先，在许多机构的规格说明文档标准中不允许数据流程图中的两个处理直接连接。因此，图8-26中需要这两个数据存储，以和标准一致。第二，这两个数据存储减少了处理“计算每周预望收入、抵押付款及津贴”与处理“计算可用资金并产生资金报告”间的连接。如图8-26所示，一旦预望收入、抵押及津贴数据被计算并存储，则一个用来计算每周可用资金的独立程序随时可能运行。简而言之，保留这样两个“无用”的文件实际上是软件工程中好的惯例；另外，通过减少两个过程间的连接可以增加最终产品的可维护性。需要数据存储“利润数据”的原因也与此类似。

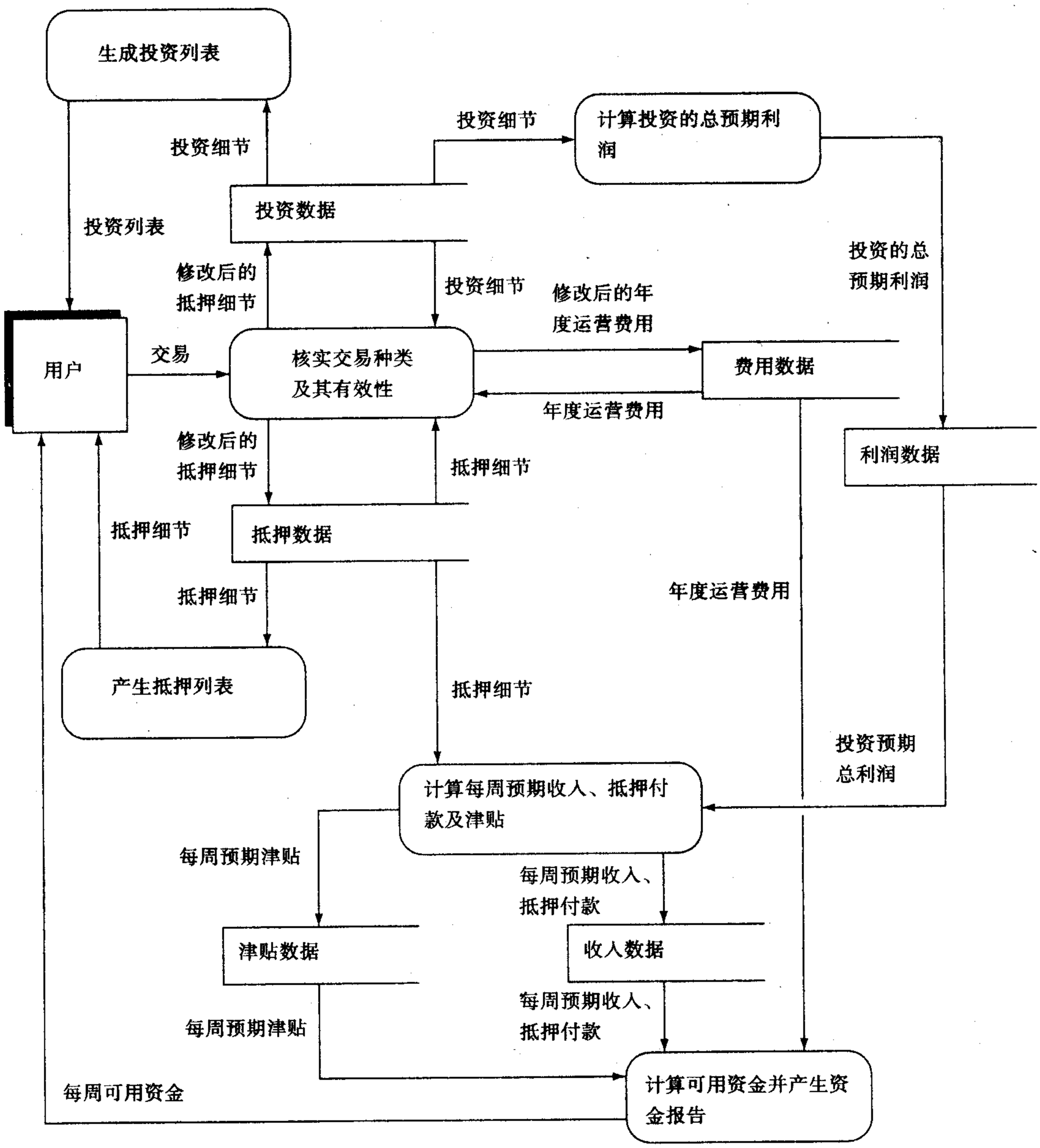

图8-26　MSG实例学习的数据流程图

附录D中资料的组织与表示法可以使用户准确快速地了解打算构造什么样的产品。然而，

达到这一理解的主要原因是用户可以在需求阶段对快速原型进行实验。

本章回顾

规格说明(8.1节)可以用非形式化(8.2节)、半形式化(8.3节到8.5节)或形式化(8.6节到8.9节)方法来表达。本章的主题是，虽然非形式化技术使用起来很容易，但它不精确；这是通过一个示例进行论证的(8.2.1节)。相反，形式化技术很有效，但在培训阶段要花大量的投资(8.10节)。在8.3节较详细地描述了一种半形式化技术，即Gane与Sarsen结构化系统分析。本章所介绍的形式化技术包括有穷状态机(8.6节)、Petri网(8.7节)及Z(8.8节)。在8.11节中介绍了规格说明的复查，然后是CASE工具(8.12 节)及规格说明阶段的量度(8.13节)。本章最后是规格说明阶段的MSG实例研究(8.14节)。

进一步阅读

关于一些特定的半形式化技术的经典作品是DeMarco所著的[DeMarco,1978]、Gane和Sarsen的[Gane and Sarsen,1979]以及Yourdon 和 Constantine的[Yourdon and Constantine,1979]。这些概念已在[Martin and McClure,1985]及[Yourdon,1989]中得到了更新。面向数据的设计(11.5节)是一个集成了不同种类的半形式化规格说明技术的设计方法，详见[Orr,1981]、[Warnier,1981]以及[Jackson,1983]。SADT(结构化分析与设计技术)参见[Ross,1985]，PSL/PSA参见[Teichroew and Hershey,1977]。有关SREM的两个信息来源分别为[Alford,1985]和[Scheffer,Stone,and Rzepka,1985]。

有关形式化技术的杰出论文主要收录于1990年9月版的《IEEE软件工程学报》、《IEEE软件》及《ACM SIGSOFT软件工程摘要》中。有特殊兴趣的读者可以阅读[Hall,1990]的全文。[Wing, 1990] 中阐述了6种形式化技术。

早期引用有穷状态机方法的的论文是[Naur,1964]，不过遗憾的是，文中将它称为图灵机(Turing machine)方法。有穷状态机方法参见[Ferrentino and Mills, 1977]以及[Linger,1980] 。[Chandrasekharan,Dasarathy,and Kishimoto,1985]中给出了一个实时系统的有穷状态机模型。关于有穷状态机在人机交互中的使用，参见[Wasserman,1985]。FSM的图形工具在[Harel,1987]中有所描述。状态图是有限状态机的一个有力的扩充，可参见[Harel,1987]和[Harel et al.,1990]。面向对象的状态图扩展参见[Coleman,Hayes,and Bear,1992]。

[Peterson,1981]中详细介绍了Petri网及其应用。Petri网在原型中的使用参见[Bruno andMarchetto,1986]。同步Petri网参见[Coolahan and Roussopoulos,1983]。

关于Z，[Spivey,1988]及[Diller,1990]中有详细介绍。有关规格说明语言详细的参考手册参见[Spivey,1992]。

实时系统的规格说明在1992年9月版的《IEEE软件工程学报》中有所论述。分布式系统的规格说明参见[Kramer,1994]。《软件规格说明与设计目标论坛学报》是有关规格说明思想研究的信息来源。

问题

8.1 为何下列约束条件不会出现在规格说明中？

(1)产品必须极大地提高全国范围内的数据收集速度。

(2)必须以合理的费用安装图形显示部件。

8.2 考虑下面的烧烤菜谱：

成分：1棵洋葱、1罐冻橙汁、新榨的柠檬汁、1/2杯面包屑、面粉、牛奶、3个中等大小的葱、3个中等大小的茄子、1个新鲜的Pockwester、1/2杯Pouilly Fuissé、1个蒜头、意大利干酪、4个鸡蛋。

前一天晚上，取来一个柠檬，挤压，出汁，并冷冻。将一棵洋葱和三根葱切成方块，在煮锅中烧烤。当冒出黑烟时，加入两杯新鲜的橙汁，用力搅拌，再将柠檬切成薄片并加入混合物中。同时，将蘑菇裹上面粉并侵入牛奶中，然后放入装有面包屑的纸袋中摇。将1/2杯Pouilly Fuissé放入炖锅中加热，当达到170度时，加入糖并继续加热，当糖变焦时加入蘑菇。将混合物加热10分钟，直到大的块子都被除去。加入鸡蛋。现取出pockwester，杀死它，剥去皮，切成大块，加到混合物中，开盖烧至沸腾。鸡蛋先用打蛋器用力搅拌5分钟。当pockwester按起来已软时，将其放入盘中，洒上干酪，烧烤不超过4分钟。

确定前面的规格说明中的模糊、遗漏及矛盾点(对于这个记录，pockwester 是一种想象的鱼)。

8.3 纠正8.2节中的规格说明段，以更准确地反映客户的需求。

8.4 用数学公式表示8.2节中的规格说明段，并将你的答案与8.3题的答案相比较。

8.5 写出产品的一个准确的英文规格说明，谓词银行声明(问题6.17)是否正确？

8.6 画出一个你在回答问题8.5时制订的规格说明的数据流程图，确保你的数据流程图只反映数据的流程，而未做出有关计算机化的假设。

8.7 考虑问题6.18的图书馆图书流通系统，写出图书馆图书流通系统的准确规格说明。

8.8 画出问题6.18中图书馆发行系统运行的数据流程图。

8.9 使用Gane与Sarsen的技术完成问题6.18 中图书馆发行系统的规格说明文档。在没有确定数据的地方(如每天出入的书目总量)自己做假设，但是要表达清楚。

8.10 一个浮点二进制数的构成是：一个可选符号(+或-)，跟上一个或多个二进制位，加上字符E，再加上另一个可选符号(+或-)及一个或多个二进制位。例如浮点二进制数11010E-1010、-100101E11101和+1E0。

更形式化的表示为：

```
<floating-point binary>    ::=[<sign>]<bitstring>E[<sign>]<bitstring>
<sign>                     ::=+|-
<bitstring>                ::=<bit>[<bitstring>]
<bit>                      ::=0|1
```

(符号[...]表示可选项，a|b表示a或b)。

假设有这样一个有穷状态机：以一串字符为输入，确定字符串中是否含有合法的浮点二进制数。试对这个有穷状态机进行规格说明。

8.11 使用有穷状态机来说明问题6.18中的图书馆图书流通系统。

8.12 如何使用问题8.11 的解答为图书馆图书流通系统设计与实现一个菜单驱动的产品。

8.13 使用Petri网说明问题6.18中一本书在图书馆中的循环过程。在规格说明中包括操作H、C及R。

8.14 假设你是一家大公司的软件工程师，专门研究计算机化图书馆系统。你的领导要求你使用Z对问题6.18中的图书馆图书流通系统做一个完整的规格说明。你将如何做？

8.15 (学期项目) 用导师所指定的技术，对附录A中所述的Osbert Oglesy产品，构造一个规格说明文件。

8.16 (实例研究)使用有穷状态机技术拟定MSG产品的需求。

8.17 (实例研究)使用Petri网技术说明MSG产品中的一次投资所经历的状态。

8.18 (实例研究)使用8.8节中的Z结构对MSG产品的一部分进行规格说明。

8.19 (软件工程文献阅读)导师将分发[Hall,1990]的拷贝，你是否同意Hall的观点，或是认为“七个神话”中至少有部分是真实的？证明你的观点。