分布式实时处理软件研究的若干问题*

刘 键 （华中理工大学）

摘 要

分布式实时处理是当代计算机科学技术前沿课题之一，它是机器人学、计算机集成制造学、并行分布式实时处理理论等边缘科学中的一个核心问题，本文较全面系统地分析了分布式实时处理系统的特征及其设计方法学中的十个基本问题，综合评述了解决这些问题的一些主要方法，并着重讨论了分布式实时处理的模型问题。

一、基本问题

所谓实时分布式计算系统，就是对来自外部物理过程的交互作用作出及时响应以达到预定目的的分布式计算系统。我们假定（或要求）该物理过程的行为可由一组事先给定的数学模型所决定，这种数学模型通常都是一组物理时间的函数。该系统接收由传感器传来的有关物理过程状态变化的信息，根据该数学模型的计算结果作出决策，再通过调节器对该物理过程的行为加以控制使得该过程的行为满足预期的要求（图1）。

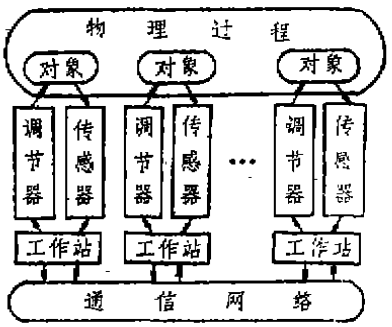

图1

实时分布式系统有着广泛的实际背景，例如计算机集成制造系统，柔性生产系统，机器人系统，飞行控制系统，作战指挥系统，防空系统等等，它与我国四化建设紧密相连。

实时计算机系统与非实时计算机系统的主要区别在于软件。分布式实时处理软件的设计是程序设计发展的新阶段。N.Wirth 曾将程序设计按复杂性和难度的高低，由低到高分成三类，即：顺序程序设计，多道程序设计与实时程序设计[24]。目前，分布式程序设计特别是分布式实时程序设计，仍处于初期发展阶段，加强分布式实时程序设计的研究，在理论上也有重要的意义。

要对分布式实时程序设计进行研究，必须先要明确实时应用问题的特征。那末，实时应用问题有那些特征呢？

1. **实时性** 包含两点涵义

·及时性：要求对来自外界物理过程的交互作用作出及时的反应。这种及时性是通过完成截止时间来描述的。超过截止时间而未完成，将可能导致灾难性的后果。例如，反导弹系统，如果不能在敌方导弹到达之前把它拦截掉，则将被敌方导弹击中。这种实时性，文献中称之为硬实时。本文主要讨论硬实时系统。

·物理时间：是现实世界的物理时间而不是人们定义的逻辑时间。

2. **开放性** 实时计算机系统与外界物理环境是交互作用着的，这种交互作用体现出实时系统的开放性，并有以下特点：

·不局部：这种交互作用将影响该实时系统的全局行为。

*本项目属国家自然科学基金委员会资助的高技术探索研究项目

·不可透明地恢复的：外界物理过程通过传感器对实时计算机系统输入一信号，这种信号是不能再恢复的。特别，如果实时计算机系统通过调节器对外界物理过程发出一命令，这种命令一经发出。例如发出一个导弹命令，就无法将这个命令取消，因为这种作用是实时的而且是物理地实现的。

·多维的：通常所指的交互作用一般仅指可能产生的行为或功能作用，并没有要求一定要在什么时间发生，而实时系统却不一样，它不但要考虑功能，而且要考虑定时。时间和错误的错误的功能一样，都会使实时系统产生严重的后果。因此，实时程序正确性验证比非实时程序验证要困难得多。

·非预期的：和计算机对其外部设备的控制不同，外界物理过程相对于与计算机系统的关系来说，是主动的，它有其自己的自然发展规律，人们不能改变这种规律，只能研究认识和巧妙地利用这种规律来达到自己的目的。但是，人们对规律的认识只能是近似的渐进的和相对的，很难说能够一次彻底认识这些规律。因此，总存在着一些未经认识的方面，从而产生非预期性。另一方面，事物发生还有它的偶然性（随机性事件），在这种情形，实时系统的交互作用当然就更具有非预期性。

3. 多样性与多变性 现实世界是丰富多采、千差万别的，即使同一部件（人工制造的东西），还会有规格型号的不同。因此，外界物理过程也是多变的。为适应这种情况，实时计算机系统必须是自适应而且是可重构的。

4. 分布性 一个外界物理过程，一般都是由多个不同功能不同规格相互协调彼此合作的部件组成的系统，而且地理上是分布的，因此，也要求实时计算机系统应是分布式系统。

总之，实时应用问题有许多自己的特点，这些特点给实时软件的开发带来许多矛盾与困难，主要有：

1. 实时系统的开放性与系统设计的封闭性的矛盾 实时系统的开放性体现在计算机系统与外界物理环境的交互作用。这个外界物理环境虽然不属于计算机系统本身，但在进行实时系统设计时，不但不能排除这个外界物理环境，而且还是系统设计的主要依据与出发点。这就是说，系统设计是封闭的。这样，系统的开放性与设计的封闭性便形成了一对尖锐的矛盾。解决这一矛盾的基本方法是：对外界物理环境及其行为给出一种假定，建立一个数学模型。这种模型有连续的，更多是离散的，有决定性的，也有随机的。这种数学模型与用户需求是我们设计的基础。

2. 规格说明的确定性与外界物理环境行为的多变性与非预期性的矛盾 这一矛盾有两种解决办法：一是根据环境的变化来修改规格说明与系统设计，从而要求系统易修改易扩充。但在实时系统情形，这种办法未必能够使人们满意。当外界环境突然变化时，对其交互作用仍然需要进行实时处理，不可能停下来等修改完设计，修改完计算机系统以后再来处理这些信息。因此，应当要有更好的办法。另一种办法，就是要使系统能够自学习、自适应、自组织。现在有的决策支持系统的设计，就设想有多个级别来进行自学习与自适应。最低层面向实时输入数据，其次面向数学模型。这是一种具有参数的可用解析式表示的数学模型。最上层面向方法论。每一层都是一闭环控制系统，较低层的偏差信息送到较高层作为修改模型或方法论的依据。例如，机器人控制系统就是这样的分级闭环控制系统。

3. 对外界物理世界认识的近似性、非精确性与数学模型要求的精确性的矛盾 有许多物理世界确实难以给出完全的、精确的描述，从而人们对它们的认识也是不完全的，非精确的。对于这种信息，人们称之为模糊信息。因此，我们需要有处理模糊信息并进行模糊推理的方法与理论。人们对外界对象的辨认，例如小孩对狗的辨认，是一个不断学习比较、不断进行特征抽取的长期认识过程。他并不需要将一条狗的所有信息进行比较，而是将这些信息进行抽象，去粗取精，去异求同，经过归纳合并分类，抽取出若干本质特征，然后将这些特征与标准模型的特征进行比较、匹配和分析，从而加以识别。所谓的模糊推理，从低层来看才是如此，实际上，在高抽象级上是根据抽象特征

与推理规则进行精确判断推理的。当然，也有另一种实现方法，就是将高抽象级的概念与推理规则加以"模糊化""插值化"，从而还原到低抽象级来考虑，所谓模糊集合理论就是这种方法。这两种实现方法，都是同一理论观点，只是实现方法有所不同，前者"自底向上"，后者"由顶向下"。这两种方法，对于设计一个类似智能机器人那样的智能实时处理系统，都是非常有用的。

4. 传感数据测不准与系统稳定性的矛盾 由于传感器件的精度限制与信号传输过程所受到的各种可能干扰，计算机系统接收到的传感数据与实际情况难免有一定偏差。因此问题在于，对于输入信号的微小扰动，系统的输出偏差是否也可以保证任意地小。这种问题通常称之为稳定性问题。当外界物理环境可以用微分方程描述时，这个问题已经研究得十分成熟了，但是，对于离散事件系统，这个问题甚至还缺乏公认的一致提法。P.J.Ramadge 与 W.M.Wonham 用状态机所产生的语言来描述外界物理过程的行为，在此基础上，他们提出适定 (Well-Posed) 监控程序与适定语言的概念。意思就是：虽然系统受到小的干扰，当相应状态仍然是可控的，不会影响设计目标的满足。与此同时，他们提出了一整套理论和方法来解决这个问题[9,10,11]，最初，J.A. Starkovic 对此也作过讨论[23]。

5. 传统离散数学模型无时间性与实时处理有物理时间性的矛盾 传统离散数学模型如状态机、Petri网，一阶谓词逻辑等都不会有物理时间，从而难以直接应用于实时处理模型的建立。一阶谓词逻辑已经扩充为时态逻辑，尽管能方便描述并行程序或分布式程序，但含有的是时序概念，而并非物理时间。和所有的谓词逻辑一样，它只便于作功能说明，而不便于作性能说明。状态机，例如前面所述Ramadge-Wonham模型，把字母表看作是事件集，于是也只能有事件先后关系而没有物理时间的概念。因此，他们提出的理论和方法很难用于硬实时系统。尽管Petri网可用来说明顺序、选择、并行与同步等概念，并可用来分析系统的安全性与活力性 (liveness)，但是，它不具有定时说明机制，而且也缺乏数据流与层次模块结构，所以也不便于描述实时处理模型。近几年来，许多人采用各种办法来克服建立实时处理模型的困难，但都未获得完全的成功。对此，后面我们还要作些简单介绍。

6. 系统容错和故障可恢复要求与实时交互过程是物理的不可透明地恢复的矛盾 实时交互过程是物理的不可透明地恢复的，现有的向后恢复的故障处理技术用不上，而必须采用向前恢复或其它容错技术。最常用的向前恢复技术是异常处理；现在的许多语言如PL/1和Ada都没有这种机制。但是，由于对可能故障的类型、原因与解决办法事先难以作无遗漏的估计，事故发生后又难以在很短时间内作出准确的诊断，所以这种方法在实时处理系统中作用不很大。比较常用的办法是采用冗余技术，其中一个典型问题是Byzantime将军问题。关于这个问题，已有许多好的解法，在实际工作中，三模块表决技术就是最常用的一种[26]。

7. 实时系统要求有唯一的全局物理时钟与分布式系统中不存在唯一的全局物理时钟的矛盾 解决这个矛盾的基本办法就是要使分布式系统中各结点拥有的局部时钟彼此同步，产生精确的容错的近似全局时间。时钟同步的基本思想是：各结点周期性地获得与参加同步的其它时钟的时差并以此作为参数，根据所选择的修正函数对自己的逻辑时钟进行修正从而达到同步。已经有许多时钟同步的策略，[21]一文对该问题作了很好的综述，并进行了分类比较。

8. 现实世界的多样性与理论模型的单一性的矛盾 实时系统所要处理的外部物理世界类型是多种多样的，有的是连续过程，有的是离散过程，有的是随机现象，还有的是决定性现象。这几种基本类型又可以组合

成更多的类型。对诸如此类特性各异的外部物理世界，不能不用相应的特性各异的数学方法来描述它们。因此，需要建立多种类型的数学模型，有些甚至无法建立精确的数学模型，而只能进行非形式的描述。

9. 实时任务的定时性与任务调度的灵活性的矛盾 实时任务何时到达，最迟要在什么时候完成，都是定死的，无多少迴旋余地。但任务调度却要以灵活性为前提，加上非周期任务的非预期性，不能不进行动态的自适应调度。这样大大地增加了调度的困难性。常用的动态调度方法有投标算法、波动算法、聚类法等等，但最关键的最基本算法是有保证的调度算法。J.Stankovic与K.Ramamritham讨论过最迟调度策略，文[18]讨论了有保证调度的一般理论，提出最小调整度优先策略，并比较了最早启动策略，最迟调度策略与最小调整度优先策略的能力。文[19]对这个问题作了进一步的研究，首先扩展了称之为RA*算法的A*算法，将RA*应用于这个问题，得到了一个最优初始分配算法。

10. 分布式系统中单一信息的局部性与用于决策信息的全局性的矛盾 这是分布式系统中普遍存在的困难，实时系统当然也不例外。在分布式实时系统中，每一子系统在一个工作站上运行。单一子系统上所接收到的传感信息，并不能对全局控制作出决策，全局控制决策一般要由所有工作站进行合作，这种合作是通过系统控制器（监控程序）来保证的。系统控制有两类，一类是集中控制，即在各工作站之上加一主控机，运行系统控制程序，这是所谓的分级控制。另一类是分布式控制，即系统中多个系统控制器执行某种协议或算法，以达到共同实现系统控制的目的。这两种控制方式，各有其特点，各有其适当的应用环境。在大型复杂系统中，一般只能采用分布式控制，而不能采用集中控制。但由于分布式协议设计的理论与方法还不成熟，故分布式控制的设计还有相当的困难。

二、模型

从上述分布式实时系统的特点可知，完全由人工来控制这种系统的设计是十分困难的，必须建立适当的形式模型及其相应的分析工具。1983年G.Le Lann在IFIP主办的世界计算机大会上的特邀报告中，提出了一个分布式实时系统开发模型。首先论及了外部物理过程特征的规范说明的结构。这个规范说明应包括物理过程中的对象，每一对象的可能状态集，不同对象的状态所应满足的约束条件，以及改变诸对象的状态的操作集。对于状态与操作，必须给出时间约束。操作与状态都可能出错，必须说明影响每一操作与对象状态的可接受的错误概率。如果要将系统分成若干个抽象级，则可用某一级的规范说明导出下一级的规范说明。每一级都要定义一抽象规范说明与设计规范说明。抽象规范说明包括抽象数据对象与函数，设计规范说明包括对象集的描述以及对于每一对象提供有关该对象的可能状态集，状态之间的关系（不变式），操作集，可记录状态集，可接受的故障概率，每一异常处理或灾难处理所要调用的操作序列以及对每一状态，每一操作的定时约束，等等。从一个抽象规范说明可导出若干个不同的设计规范说明。

一个分布式实时程序系统由若干个模块组成，每一模块包含一个对象和一同步器，对象的描述如上所述，同步器的职责是：

- 检查对一操作的外部请求是否正确
- 检查一请求操作的执行时间是否超过所指定的时间
- 检查是否出现异常或灾难（并初始化特定操作）
- 如有必要，对每一操作施加一原子性质
- 对每一对象施加一合法操作序列（满足指定的不变式）

这个模型虽然还不是一个形式模型，但很容易将其中关于对象的不变式、约束条件、状态操作的描述等形式化，若有适当的分析方法支持，就可成为一实时系统的形式模型。

注意，模型特征通常是通过系统的规范说明来体现的，而规范说明则要通过某种规范说明语言来表示，这两者紧密相关。因此，在下面的讨论中，有时谈的是模型，有时则论及规范说明语言。

对于建立分布式实时处理模型，通常是以状态机、Petri网、时态逻辑为基为，再加上一些性能约束（例如定时约束）机制。这些方面虽然近年来提出了不少建议，但总的来说还是不成熟的。下面介绍几个例子。

加州大学Santa Barbara分校设计了一个实时系统规范说明语言RT-ASLAN，它是顺序程序规范说明语言ASLAN的一个扩充。ASLAN规范说明语言建立在一阶谓词演算之上，并使用状态机方法进行验证。被说明的系统可以看成为依赖于状态变量值的各种状态。仅当通过良定义的转移时才可能发生状态的改变。每一状态以及两个相继状态间必须使具有所希望性质的断言成立。每一状态都必须满足的断言叫不变式，两相继状态间的断言叫做约束。为保证规范说明满足不变式与约束断言，ASLAN生成一些候选引理，以备构造关于该规范说明的正确性证明，这些候选引理被称为正确性猜想。一个ASLAN规范说明由一系列抽象级组成。第一级是该系统的抽象模型，说明由什么组成（类型、常量、变量），要做什么（状态转移），要满足那些关键性需求（不变式、约束）。较低层是较高层的细化，用实现语句说明。

RT-ASLAN是ASLAN的扩充，主要增加了排序操作符与接口转移。用RT-ASLAN模拟的实时系统可看成由通信接口进行通信的若干个进程组成。每一进程通过一个类似于ASLAN规范说明的通信子规范说明来表示。在同一通信子规范说明内的转移是串行的，但在不同的通信子规范说明中的转移则是并行执行的。在每一通信子规范内，转移可以是周期的，或者是外部的。和通常的实时系统的观点一样，这里认为只有外部转移才对应于现实世界事件。

实时系统规范说明的特点是性能猜想，RT-ASLAN实现正确性猜想的基本思想是：通过级间正确性猜想来保证功能性，并且又产生性能猜想以证明每级抽象是可行的，证明较低层满足较高层的定时约束。性能猜想可以分成可行性猜想与实现猜想两类。可行性猜想又有两种：第一种是定时约束，可写为

$$T'_{入口} \xrightarrow{\leqslant C_T} T_{出口}$$

其中C_T为T的执行时间上界，$T_{入口}$为T的入口条件，表示T要施加作用的状态，$T_{出口}$是T的出口断言，表示T作用完毕时的状态。这个猜想的意思是：若T被应用，则其应用时间最多为C_T。第二种可行性猜想是全局可行性猜想，它表示对每一规范说明，与该说明相联系的处理机不超过负荷。实现猜想是要断定：给定一高层周期时间C_T，不管选择哪一可能的实现转移，较低层转移的作用时间不能超过C_T。RT-ASLAN性能猜想可从以下例子得到解释。注意，生成性能猜想所需的唯一信息是最顶层的周期时间，排序操作符与实现语句。所有的功能信息都从例子中删去了。

```
LEVEL Top-level
TRANSITION Upper CYCLE 10
EXIT
  DO S1,1 BEFORE S1,2 OD
END Top-Level
LEVEL Second-Level
TRANSITION L2,1
EXIT
  DO S2,1 BEFORE S2,2 DO
TRANSITION L2,2
      ...
IMPLEMENTATION
  Upper==DO L2,1 BEFORE L2,2 OD
END Second-Level
LEVEL Third-Level
TRANSITION L3,1 TIMING 2
      ...
TRANSITION L3,2 TIMING 3
```

...

TRANSITION $L_{3,3}$ TIMING 2

...

IMPLEMENTATION

$L_{2,1}$==DO $L_{3,1}$ BEFORE $L_{3,2}$ OD,

$L_{2,2}$==$L_{3,3}$

END Third-Level

现在我们进行逐层分析，并对顶层中的每一周期性转移建立一个性能表达式（PE）。首先考虑顶层中的转移Upper。初始化时，PE为TRUE。显然，Upper必须要在它再次应用之前完成，即

$$PE \leftarrow M_{Upper} \leqslant C_{Upper}$$

其中M_T为T的最大作用时间，"←"可读作"变成"。

再从Upper的转移头部，$C_{Upper}=10$，有PE变成

$$PE \leftarrow PE \,\&\, C_{Upper}=10$$

由第二层的IMPLEMENTATION语句，可知Upper的最大作用时间是两个低层转移作用时间之和，于是

$$PE \leftarrow PE \,\&\, M_{Upper}=M_{L_{2,1}}+M_{L_{2,2}}$$

最后看第三层，转移具有TIMING属性，这表示它们的绝对执行时间不超过TIMING值。由IMPLENTATION

$$PE \leftarrow PE \,\&\, (M_{L_{2,1}}=M_{L_{3,1}}+M_{L_{3,2}}) \,\&\, (M_{L_{2,2}}=M_{L_{3,3}})$$

假设$M_{L_{3,1}}=2$，$M_{L_{3,2}}=3$，$M_{L_{3,3}}=2$，有

$$\begin{aligned} PE= & \\ & M_{Upper} \leqslant C_{Upper} \\ & \&C_{Upper}=10 \\ & \&M_{Upper}=M_{L_{2,1}}+M_{L_{2,2}} \\ & \&M_{L_{2,1}}=M_{L_{3,1}}+M_{L_{3,2}} \\ & \&M_{L_{2,2}}=M_{L_{3,3}} \\ & \&M_{L_{3,1}}=2 \\ & \&M_{L_{3,2}}=3 \\ & \&M_{L_{3,3}}=2 \end{aligned}$$

化简后，PE等于$7 \leqslant 10$，即永真，从而证明该实现满足顶层给出的性能需求。

RT-ASLAN有其特色和优点：①提供一些定时说明机制，如DO-BEFORE-OD，TIMING，等，以及性能猜想说明，这对于定时性的说明与验证是很有好处的，②整个结构类似于CSP的结构，每一通信进程建立在状态机与一阶谓词基础上，在一定程度上便于程序的功能验证。但是，作为一个分布式程序，仅仅验证各通信进程是远远不够的，必须要有各进程证明合作性的验证。关于这点，语言中没有提供任何有力手段。其次，虽然提供一些定时说明手段，但是非常低级。这必然要求程序员进行复杂的实时任务调度，很不合理。理想的情况是，只要求程序员对每个实时任务的到达时间、计算量以及截止时间作出说明，其余工作（如初始分配与静态调度）由系统来完成。若调度不成功，则显示出错信息，要求系统设计员进行处理。再次，将功能验证与性能验证分别进行，在通常情况下，也许是行得通的。但是，在实时系统中，功能和定时是密切相关的。例如，一个反导弹程序，分成许多部件，虽然从功能上说，能够准确地击中目标，但是某些部件在规定的定时里不能完成计算，从而整个程序不能在敌方导弹袭击我方之前运行完毕，反导弹功能又有何可言呢。

此外，德克萨斯大学的F·Jahanian与A，Ka-Lau Mok提出了实时逻辑RTL。他们关于实时系统的模型（即事件-动作模型）很简单。除了事件和动作以外，规格说明中还增加了状态谓词与定时约束。动作允许嵌套、串并组合，但不允许出现循环。他们关于系统安全性的分析，基本方法步骤如下：①根据事件-动作模型写出系统的规格说明，②将规格说明转换成RTL，即用实时逻辑RTL来表示规格说明，③用RTL写出安全性断言，④利用演绎系统证明该断言是否与说明一致。他们的证明方法不是利用Hoare逻辑来寻求不变式，而是以定理证明方法为基础，先作出系统说明与断言的合取，并求该合取式的否定式，然后证明不存在任何如下定义的出现函数（用于描述定时）能够满足该否定式。他们提出了实时逻辑RTL，为各种不同类型的约束条件提供一个统一的描述方法，并为这种实时逻辑，寻求了一种证明其永真公式的形式方法。现简单介绍如下。

实时逻辑的核心概念是出现函数@，它是(E,

W)→W的一个映象，其中E是事件常数集合，W是非负整数集合。@(e, i)(e∈E, i∈W) 的意思是表示事件e的第i次出现的时间。利用这个函数，可以形式地描述各种类型的约束。例如

实时活动开始时间与结束时间有如下关系

$\forall t_1\forall t_2\forall i[@(\uparrow A,i)=t_1\wedge@(\downarrow A,i)=t_2\wedge t_1\leqslant t_2]\rightarrow]\rightarrow t_1+C(A)\leqslant t_2$

其中↑A表活动A开始事件，↓A表活动A完成事件，C(A) 表A的执行时间。

组合活动约束，设A=B, C表示A由B和C串行组合而成，先B后C, $A'=B'|C'$ 表示A′ 由B′ 和C′并行组合而成，则有

顺序约束 $\forall_i@(\downarrow A\cdot B, i)\leqslant=(\uparrow A\cdot C, i)$

并行约束 若A=B; (C|D)，则

$\forall_i@(\downarrow A\cdot B, i)\leqslant@(\uparrow A\cdot C, i)$

$\forall_i@(A\cdot B, i)\leqslant@(\uparrow A\cdot D, i)$

偶发性活动约束 所谓偶发活动A是指：当触发事件E发生时，就执行A，其截止时间为d，最短间隔为p。用RTL表示，就是：

$\forall i\exists j@(E, i)\leqslant@(\uparrow A, j)\wedge@(\downarrow A, j)\leqslant@(E, i)+d,$

$\forall i@(E, i)+P\leqslant@(E, i+1)$

周期性活动约束 所谓周期性活动A是指：当状态S为真时，执行A，其周期为p，截止时间为d。设状态谓词S(x, y) 表示状态S在时刻x变为真，在时间区间(x, y) 内保持为真，仅当时刻y或y以后才可能变为假。于是周期性定时约束用RTL可表示如下：

$\forall x\forall y\forall n\exists i\ \ S(x, y)\wedge y-x>n*p\rightarrow x+n*p\leqslant@(\uparrow A, i)\wedge@(\downarrow A, i)\leqslant x+n*p+d$

由以上数例可知，出现函数@的定时描述功能是很强的。RTL公式就是由整常量、事件常量、整变量、具有整变量下标的事件或活动常量、出现函数、值域为整数集合的未解释函数、加减函数、乘以常数的函数、等式/不等式谓词、表示一时间区间内某一状态的真值的状态谓词等成分加上全称量词、存在量词与一阶逻辑连接符所组成。它仍然是一阶谓词逻辑。但由于其中包含有出现函数@(e, i) 与线性不等式，用传统的逻辑演绎系统就难以处理。这个问题类似于Presburger算术逻辑，所谓Presburger谓词公式是由整常量、整变量、加法函数、等式/不等式谓词以及一阶逻辑连接符所构成。但是RTL中含有事件常量与活动常量，这是Presburger算术所不允许的。Shostak将无量词的Presburger逻辑加以扩充使其包含有未解释函数。Bledsoe与Hines又提出一个以消解法为基础的一般线性不等式的定理证明方法，于是只需从RTL中消除量词，状态谓词与非整型常量(归根到底只需消除存在量词与发生函数@(e, i))，便可解决问题。这样所有被存在量词约束的变量都可换为Skolem常数或函数，要消除发生函数@(e, i)，只需对每一事件常量e，定义一个W→W的未解释函数$f_e(i)$，其中W是非负整数集合。

总的来说，RTL确是一实时逻辑的形式系统，既能表示功能约束又能表示性能约束。尽管这种性能约束暂时还只是定时约束，但似类方法可以推广到其它性能约束，从这种意义上说，这项工作是有突破性的。但是，要使这项技术（RTL及其证明工具）实用化，还有许多工作要做。第一，同所有的定理证明的消解过程一样，这种方法是低效的，要证明一个复杂系统的安全性是不实际的。第二，要将一复杂系统划分成若干简单的子系统，或者该系统就是一个分布式系统，并且要将复杂系统的证明归结为简单子系统的证明，一个关键的技术就是类似于K.Apt提出的合作性证明。虽然在temple逻辑或推广的Hoare逻辑中，已奠定了这种合作性证明理论，但具体证明方法的效率很低，而对于定理证明理论来说，或对消解理论来谈，这种分布式合作性证明的相应理论是什么，至今还不大清楚。

总之，上述的两种实时系统的说明语言RT-ASLAN和RTL，前者和当前的实时系统说明语言例如Ada相比，没有实质上的进展；后者虽然证明的效率还要努力改进（和逻辑程序设计或定理证明的情况一样），但确是一形式逻辑，具有描述定时约束的功能。除此以外，加拿大多伦多大学P.J.Ramadge和W.M.Wonham等人利用形式语言与自动机对离散事件系统作了一系列的研究，得到了很好的结果。遗憾的是，该模型似乎不能包含硬实时系统。关于分布式实时系统的形式说明方法，还需要继续深入研究。

参考文献

[1] R.L. Glass, Real-Time Software, PRENTICE-HALL, 1983

[2] G. Le Lann, "On Real-Time Distributed Computing" INFORMATION PROCESSING 83

[3] J. Stankovic, et al, "A Review of Current Research and Critical issues in Distributed System Software" IEEE Technical Committee NEWSLETTER, Vol.7, №1, March, 1985

[4] J. A. Stankovic, "Misconceptions About Real-Time Computing" IEEE COMPUTER Oct., 1988

[5] B. Averbuheimer, R.A. Kemmerer, "RT-ASLAN, A Specification Language for Real-Time Systems" IEEE Trans. on Sofeware Engg. Vol. SE-12, №9, Sept. 1986

[6] F.Jahanian, A.K. Mok, "Safety Analysis of Timing Properties in Real-Time Systems" IEEE Trans. on Sofeware Engg., Vol. SE-12, №9, Sept. 1986

[7] Luqi, V.Berzin, "Rapidly Prototyping Real-Time Systems" IEEE Software, Sept. 1988

[8] M.S.Deutsch, "Focusing Real-Time Systems Analysis on User Operations" IEEE Software, Sept. 1988

[9] F.Lin, W.M.Wonham, "On Observability of Discrete-Event Systems" INFORMATION SCIENCES 44.1988

[10] Y.Li, W.M.Wonham, "On Supervisory Control of Real-Time Discrete-Event Systems" INFORMATION SCIENCES, 46, 1988

[11] F. Lin, W.M. Wonham, "Decentralized Supervisory Control of Discrete-Event Systems" INFORMATION SCIENCES, 44, 1988

[12] G.M.Reed, A.W.Roscoe, "A Timed Model for Communicating Sequential Processes" Proc. ICALP 86, Springer LNCS 226, 1986

[13] W. Zhao et al, "Scheduling Tasks with Resource Requirements in Hard Real-Time Systems" IEEE Trans. on Software Engg. vol SE-13, №5, May 1987

[14] K.Ramamritham, W.Zhao, J.A.Stankovic, "Meta-Level Control in Distributed Real-Time Systems" IEEE CH2439-8/87, 1987

[15] W.W.Bledsoe, "The SUP-INF method in Presburger arithmetic," Dep. Math., Univ. Texas at Austin, Memo ATP-18, Dec. 1974

[16] W.W.Bledsoe, L.M.Hines, "Variable elimination and chaining in a resolution-based prover for inequalities" 5th Conf. Automated Deduction, Springer LNCS, 1980

[17] R.E. Shostak, "A practical decision procedure for arithmetic with function symbols", J.ACM, Vol.26, №2, 1979

[18] 刘键, "分布式实时系统动态任务调度的一有效算法"《计算机学报》, Vol.10, №11, 1987

[19] 郑勇, 刘键, "分布式实时系统动态任务初始分配的有效方法", 《计算机学报》, Vol.12, №5, 1989

[20] 郑勇, 刘键, "分布式系统中实时任务的最佳初始分配"《计算机学报》Vol.12, №5, 1989

[21] 陈性元, 李平文, 刘键, "分布式实时系统中的时钟同步"待发表

[22] 刘键, 分布式计算机系统, 人民邮电出版社, 1989

[23] J.A.Stankovic, "Stability and Distributed Scheduling Algorithms" IEEE TRANS. Software Eng., Vol SE-11, №10, 1985

[24] N. Wirth, "Toward a Discipline of Real-Time Programming" CACM Vol 20, №.8, Aug, 1977

[25] F.B. Schneider, L.Lampert, "Paradigms for distributed Programs" from Chapter 8 in Distributed Systems, Springer-Verlag, 1985