浙江大学

硕 士 学 位 论 文

论文题目 基于离散事件动态系统的故障诊断理论的研究

作者姓名 吴旋

指导教师 颜文俊

学科(专业) 控制理论与控制工程

所在学院 电气工程学院

提交日期 2002 年 2 月 28 日

Research on Fault Diagnosis Theory based on DEDS

By

Wu Xuan

Supervised by: Yan Wenjun

A Dissertation Submitted to the Academic Committee

Applying for the Degree of M. Sc in Control Theory and Engineering,

Institute of Electrical Engineering

Zhejiang University, Hangzhou, P. R. China

Feb 28, 2002

摘 要

随着科学技术的发展，生产系统规模日益扩大，结构日趋复杂，而且出现了许多“人造系统”，故障的发生也日益频繁。由于现代社会对系统可靠性的要求日益提高，故障诊断就显得尤为重要。但是许多系统特别是复杂系统中存在着大量具有异步性、离散化和事件驱动等特点的故障，使得传统的故障诊断方法难以有效进行。系统的复杂性以及故障的事件化特点更适合离散事件动态系统（DEDS）的概念来处理，而 DEDS 各个理论经过较长时间的发展，已经趋于成熟，所以基于离散事件动态系统的故障诊断方法就应运而生。虽然这类故障诊断方法出现不久，还有大量问题需要解决，但已经取得了丰富的成果。而且，基于离散事件动态系统的故障诊断方法已经广泛应用于计算机和通信网络、电力系统、集成电路制造、柔性生产线、间歇生产过程、多机器人的工作单元和高速公路的智能调度等领域。

基于 DEDS 的故障诊断方法是定性的方法。由于对 DEDS 研究是多层次的，有许多模型方法；所以基于 DEDS 的故障诊断方法也多种多样。本文以 Sampath 等人提出的诊断器理论为主线，对 DEDS 故障诊断理论的特征、性质、发展和应用作了比较深入的研究。文中的主要贡献有：

1. 全面、系统地阐述了 DEDS 故障诊断理论的产生原因和发展过程，对目前 DEDS 故障诊断的众多有代表性的方法做了简要的介绍，并对 DEDS 故障诊断的优点和意义进行了阐述。而本文的研究是以监控器理论和诊断器理论为基础的，所以文中对这两个方面进行了比较详细的介绍。此外，我们也对 DEDS 故障诊断理论的应用做了一些讨论。
2. 在文中所介绍的理论和方法的基础上，我们进行了一系列研究工作，并取得了如下成果：

（1） 在监控器理论和分散监控方法的基础上，我们提出了用于全局目标分散监控的联合正态理论，该理论能较好的解决分散监控的实现方面的问题，而且对 DEDS 故障诊断理论研究也有一定的作用。

（2） 活跃诊断方法可以解决不可诊断系统的诊断问题，协调分布式诊断方法则解决具有信息分散特征的复杂系统的诊断问题。我们将分散监控、活跃诊断以及分散式诊断的结合起来，提出了分散活跃诊断的方法，较好的解决了原本不可诊断的信息分散系统的诊断问题。

（3） 我们还提出了一种针对一类大规模分布式系统的模块化诊断方法。该方法基于诊断器理论的，其模块化特征不但使得诊断器的构造不用考虑整个系统的情况，而且大

大减少了系统构成的变化对已构建好的诊断器的影响。

3. 在论文的最后，我们对本文工作进行了总结，并介绍了 DEDS 故障诊断领域特别是监控器理论这方面所存在的一些悬而未决的问题，最后对 DEDS 故障诊断领域的研究和应用的前景进行了讨论。

关键词：故障诊断，离散事件动态系统（DEDS），诊断器，监控器，控制诊断器，活跃诊断，分布式系统，信息分散系统，分散监控，模块化诊断，联合能观，联合正态。

Abstract

The scale of production systems becomes larger and the structure becomes more complicated with the development of science and technology. And many "manufacture systems" appears. Under this condition, faults occur more continually. Fault diagnosis becomes more important with the improving request of system reliability. But in many systems especially complicated systems a great deal of fault with the character of asynchronies, discrete and event-driven exist. These faults cannot be effectively diagnosed by traditional diagnosis method. The notion of discrete event dynamic systems (DEDS) is more adaptive because of the complexity of systems and event-driven character of fault. Having been developing for a relative long time, theories of DEDS are put forward. In this way, fault diagnosis methods based on DEDS come into being. A large number of results have been get although these methods are still young and many problems are still in the scale. Furthermore, these methods have been widely applied in many fields, such as networks of computer and communication, power system, IC manufacture, flexible manufacture, intermittent producing process, working units of multi-robot, intelligent attemperment of freeway, and so on.

Methods based on DEDS belong to quantitative method. Researches on DEDS use many methods, which are based on models of different levels. Accordingly, there are numerous fault diagnosis methods based on DEDS.

This dissertation concentrates on diagnoser theory initialized by Sampath, and discusses in relatively detail the features, properties, development and applications. The main contributions of the dissertation are as follows:

1. First of all, it gives an overview of the emergence and the development process of fault diagnosis theories based on DEDS. Many results of these theories are also introduced. And the advantages and research meaning of fault diagnosis based on DEDS are expatiated. Furthermore, we describe supervisory theory and diagnoser theory in detail because the two theories are the base of this dissertation. Some discuss about applications of fault diagnosis theory based on DEDS is also in the dissertation.
2. On the base of those theories and methods introduced in the dissertation, some researches have been carried out and the results gotten are showed as follows:

(1) Based on supervisory theory and decentralized supervisory methods, we present a theory of united-normality, which is used for decentralized supervisory aimed at global objective. This theory is helpful to solve some problems about the application of decentralized supervisory and researches of fault diagnosis theory based on DEDS.

(2) Active diagnosis method can be used to diagnose systems that are not diagnosable, while the specialty of coordinated decentralized fault diagnosis method is diagnosis of complicated systems with the character of information decentralize. So we present a fault diagnosis method. Decentralized supervisory, active diagnosis and decentralized diagnosis are all integrated into this method. Having those theories and techniques, this method can effectively solve the diagnosis problem of those information-decentralized systems that are not diagnosable.

(3) We present a modular approach for diagnosis in large distributed systems. This approach is based on the diagnoser theory. Because of modularity, the whole system need not be thought of when diagnosers are constructed or used and changes of components in the system have a lot smaller effect on existing diagnosers.

3. At the end, we summarize the whole dissertation. And some problems in the scale about the realm of fault diagnosis based on DEDS are introduced. Finally, we discuss the researches and applications foreground of this realm.

Key Words: Fault Diagnosis, Discrete event dynamic systems (DEDS), Diagnoser, Supervisor, Control Diagnoser, Active Diagnosis, Distributed Systems, Information-Decentralized Systems, Decentralized Supervisory, Modular Diagnosis, United Observability, United Normality.

致 谢

在论文完成之际，我衷心感谢我的导师颜文俊博士后。他严谨的治学态度，开阔活跃的思维，敏捷深刻的洞察力和良好的记忆能力都给我留下了深刻的印象。作为他的学生，我受益匪浅。他对作者的选题和科研工作都基于了充分的肯定和指导，并对作者的工作、学习和生活等方面帮助很大。这一切，我都将铭记在心。

在开展研究和撰写论文的过程中，得到了诸静教授，姚维博士，高锋副教授等老师的指导和帮助，在此向他们表示深深的感谢。在攻读研究生期间，我还得到了很多老师的教诲和帮助，在这里对他们表示诚挚的谢意。

我还要感谢实验室里的所有师兄弟姐妹、我的室友以及工自 99 研的所有同学们。对于大家的关心、帮助以及友谊，我无法忘怀。

对于文中被引用的文献的作者，在此表示深深的敬意。

总之，我要感谢所有关心爱护我的朋友和亲人。

谨以此文献给远方的父母！

吴旋

2002 年 2 月于求是园

目 录

摘要
Abstract
致谢

第一章 综述 1
1.1 研究领域简介 1
1.1.1 故障诊断 1
1.1.2 DEDS 故障诊断 2
1.2 本文研究的意义 7
1.3 全文概貌 8

第二章 DEDS 监控理论 10
2.1 引言 10
2.2 监控理论基础 11
2.2.1 建模方法 11
2.2.2 能控性和能观性 13
2.2.3 监控器 15
2.2.3.1 监控器基础 15
2.2.3.2 监控器综合 17
2.3 分散监控 18
2.3.1 基于局部控制目标的分散监控 18
2.3.2 基于全局目标的分散监控 20
2.4 用于分散监控的联合正态理论 22
2.4.1 联合正态 22
2.4.2 分散监控问题求解 25
2.4.3 例子 27
2.5 小结 29

第三章 DEDS 故障诊断理论 30
3.1 引言 30
3.2 诊断器方法 31
3.2.1 系统模型 31
3.2.2 基本概念 31
3.2.3 可诊断性 32
3.2.4 诊断器 32
3.2.5 可诊断性的充要条件 34
3.2.6 在线诊断 34
3.2.7 例子 35
3.3 活跃诊断 36
3.3.1 非有效(live)系统的可诊断性 36

3.3.2 诊断器 G_d^{live} 37
3.3.3 活跃诊断问题 38
3.3.4 例子 41
3.4 小结 44

第四章 信息分散系统的故障诊断 44
4.1 引言 44
4.2 协调分布式故障诊断方法 44
4.2.1 协调分散结构 44
4.2.2 可诊断性定义 46
4.2.3 分散式诊断的实现目的 47
4.2.4 协议 1 描述 47
4.2.4.1 本地点诊断信息 47
4.2.4.2 通信规则 49
4.2.4.3 决策规则 49
4.2.4.4 诊断属性 51
4.2.5 讨论 52
4.2.5.1 扩展到多个站点的情况 52
4.2.5.2 发生通信时延的情况 52
4.2.6 例子 52
4.3 分散式活跃诊断方法 55
4.3.1 问题描述 55
4.3.2 离线诊断 55
4.3.3 在线诊断 58
4.3.4 例子 59
4.4 小结 61

第五章 一类分布式系统的模块化诊断方法 63
5.1 引言 63
5.2 模块化诊断 64
5.3 基于诊断器理论的模块化诊断 65
5.3.1 系统模型 65
5.3.2 模块化诊断器的构造 65
5.3.3 诊断条件 67
5.3.4 在线诊断 68
5.4 小结 69

第六章 DEDS 故障诊断理论的应用 70
6.1 简介 70
6.2 诊断器方法进行故障诊断的常见步骤及应用 70
6.2.1 诊断体系结构和诊断过程 70
6.2.2 系统建模 72

6.2.3 例子与分析 73
6.3 活跃诊断应用举例 76
6.4 小结 78

第七章 结束语 79

参考文献 83

附录 A SCPO 问题 92

附录 B 监控器理论中相关子语言的算法 94
B.1 给定语言的能控子语言的上确界 94
B.1.1 当给定语言不闭时 94
B.1.2 当给定语言闭时 95
B.2 给定语言的能控子语言的下确界 96
B.3 完全能观时的其他语言类 97
B.4 部分能观时的语言类 98

附录 C 模块化监控 100

附录 D 作者攻读硕士研究生期间完成的论文 102

第一章 综述

1.1 研究领域简介

1.1.1 故障诊断

在所有的系统特别是大型或者复杂的系统中，经常会发生各种故障。而随着现代社会对于系统的可靠性、有效性和可维护性要求的日益迫切，人们研究出了许多故障诊断的方法。可靠性工程、控制理论以及计算机科学的文献中也广泛的涉及到故障诊断问题。所以，故障诊断技术是当今的一个研究热点，它包含了故障的检测、分离和辨识[92]~[96]。目前常见的故障诊断方法有下面几种：（1）基于解析冗余的故障检测和分离（FDI）方法[97, 98]；（2）基于非模型的方法，比如基于统计假设检验和信号分析（详见【99】及其参考文献）；（3）故障树方法[101, 102]；（4）基于人工智能和基于模型的推论方法[103, 104, 105]；（5）离散事件系统方法[65]~[91]。文献【50，63，100】对这些方法进行了概述。在这些方法中，解析冗余方法和非模型方法主要基于对信号的定量分析，并以差分和微分方程作为对象的研究模型；而人工智能和离散事件系统方法则是基于定性推理。

然而，在现代工业大生产、自动化生产领域和大量的人造系统中，故障不仅具有定量和时序的特点，还具有异步性、离散化和事件驱动的特征，如阀门的开关、电机的启/停、位置的定位、传感器或执行器的失灵、通讯的中断等，传统的故障诊断理论难以进行有效的处理。因此，研究一种新型的、针对离散化故障事件的理论方法和体系，是十分必要的。

目前，针对具有上述特性的故障而发展起来的一类方法主要是基于知识的故障诊断技术。该方法通过由定性微分方程构成的约束模型，采用知识观测器对故障进行诊断。但该技术在设计知识观测器需要大量先验知识，且计算繁琐，尤其不适应于大系统的故障检测，同时缺少系统化的理论方法。

这类故障的特点特别适合于使用离散事件动态系统（Discrete Event Dynamical Systems，DEDS）来表示。所以自 1992 年以来，离散事件动态系统的故障诊断技术的发展引起了广泛关注[65]~[91]。

1.1.2 DEDS 故障诊断理论

基于离散事件系统的诊断理论很多，且主要以离散事件系统的逻辑层次模型为基础进行研究。离散事件系统的逻辑层次模型主要有两种：自动机模型和Petri网模型。自动机模型方法还分为基于事件和基于状态的。所谓基于状态的方法即系统的诊断性能与可观事件的划分及所采用的自动机结构有关；而基于事件的方法则使得系统的诊断性能不依赖于具体的结构及可观事件的划分。

在这些理论中，最有代表性的 DEDS 诊断理论是诊断器理论。该理论由 Sampath 等人在 1994 年提出的[49]。这一理论延用了 Ramadge 等人提出的监控理论中描述 DEDS 的方法（即有限自动机模型），提出了可诊断性理论和诊断器的概念。在监控理论中，通常将系统的动态行为看成一连串事件的组合，每个事件用一个字符表示，则系统的行为就是一系列字符串的集合，即语言。而诊断器理论认为，如果一个语言是可诊断的，那么可以在有限的延迟内检测到特定的可辨别的不可观事件，亦即故障事件；而系统只有是可诊断的，诊断器才能对每个故障进行检测和分离。诊断器根据系统的自动机模型产生，可以在线观测系统行为，并执行诊断功能；通过查看包含故障信息这些状态，可以检测到故障发生。诊断器不但可以在线检测是否有故障发生并判定故障的类别，还可以离线确认系统的可诊断性。诊断的原理如图 1.1 所示。由此可见，该方法是基于事件的。本论文中所讨论的故障诊断方法基本上都是以诊断器理论为基础的。

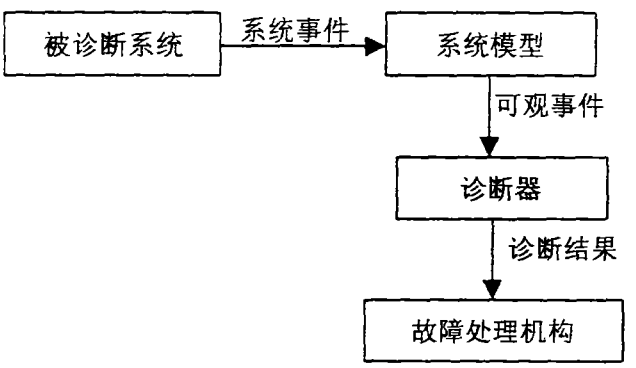


图 1.1 诊断器理论的诊断原理图

而基于状态的故障诊断理论中比较有代表性的是 F. Lin 于 1994 年提出的基于状态的诊断方法[44]。该方法包含离线诊断和在线诊断。离线时，系统处于测试状态，诊断过程发出一系列测试命令，观察结果的输出，然后在系统可能进入

的所有状态中进行推断。在线时，系统处于正常操作；诊断的目的就是通过发出一系列命令，唯一地确定系统状态的划分。与离线诊断不同的是，在线诊断过程中还必须处理其他不可控事件可能会发生的情况。专门有算法可以得出测试命令序列，且该算法当系统是在线可诊断的情况下是收敛的。Bavishi 和 Chong 还提出了离散事件系统可测试理论即两个算法[47]：（1）用最少的传感器来保证给定系统可测试。（2）给定传感器集合，决定保证系统可测试所需的最少状态。

事实上，即便对于连续动态系统，在许多情况下，人们更关心的是它的一些定性信息而非它的精确值，如料位、温度等变量的高、中、低。为此，可引入一个定性化系统，并根据工艺或知识和设计要求，将系统动态行为转换成一串时间序列或事件序列，从而将连续系统的故障诊断问题转化为离散事件系统的故障诊断问题。Lunze 等对该类问题进行了研究并成功地应用于间歇生产过程中[86]。

很多系统的诊断部分通常都是一个独立的模块，而开始设计系统的时候通常没考虑可诊断性因素，所以控制器设计和诊断器设计是分离的，这样必然影响到系统的诊断性能。于是 Sampath 等人于 98 年提出一个能同时进行控制和诊断的整合方法，即活跃诊断[53]。该方法以 DEDS 为框架，并充分利用了 DEDS 诊断领域及部分可观的监控器控制方面的成果。他们将活跃诊断问题表示成一个监控器控制问题，其合法语言是系统行为的正则语言形式的一个“适当”的正则子语言。他们还提出一个迭代过程以产生合法语言的可控、可观且可诊断的子语言上界并得到相应的监控器。这种方法不但提供了一个保证闭环系统具有可诊断性的控制器，而且也提供了供在线故障检测的诊断器。活跃诊断的原理如图 1.2 所示。这样，基于 DEDS 的诊断理论就可以更好地吸收监控器研究领域的最新成果，并能更方便的运用于实际生产过程和复杂系统中。

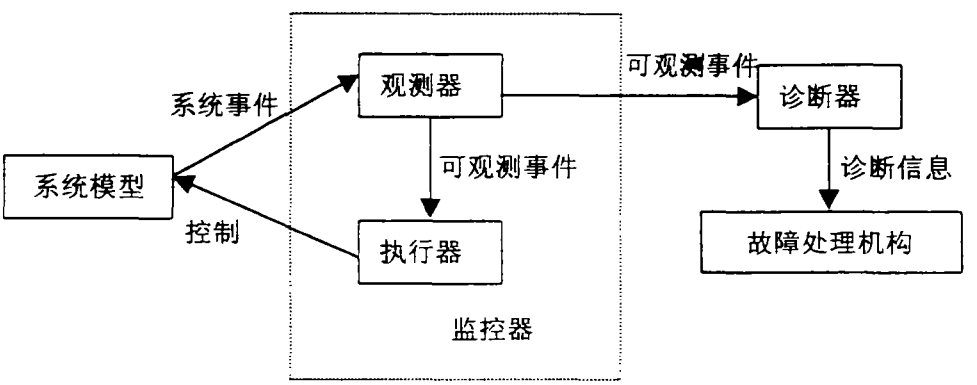


图 1.2 活跃诊断的原理图

在各类复杂系统中，最常见的就是分布式系统。离散事件动态系统理论对于分布式系统的研究有很大的作用，很多学者对此进行了研究[32]~[37]。基于这些研究成果，对于分布式系统的诊断问题引起了广泛的关注[61, 63], [65]~[70]。

一类具有代表性的分布式系统就是信息分散系统。在现实生活中，许多系统都是分散的，比如大多数技术复杂系统（计算机和通信网络，制造业，过程控制，远程控制多机器人系统和电力系统等）是信息分散的。在分散信息系统中有几个工作站（比如决定生成器、控制器和诊断器等），每个工作站都只能取得当地的信息，但可以和其他站点通讯，并交换有限的信息。既然信息是实时交换的，且交换通道容量有限，就可能存在传输延时，甚至出现故障和传输错误。此外，每个站点可得到的信息是不完全的，延迟的，甚至可能有错误。因此，对于分布式系统，其故障除了常规的元部件故障、传感器故障和执行器故障外，还包括通讯故障。所以常规的故障诊断方法不能直接用于信息分散系统的诊断。

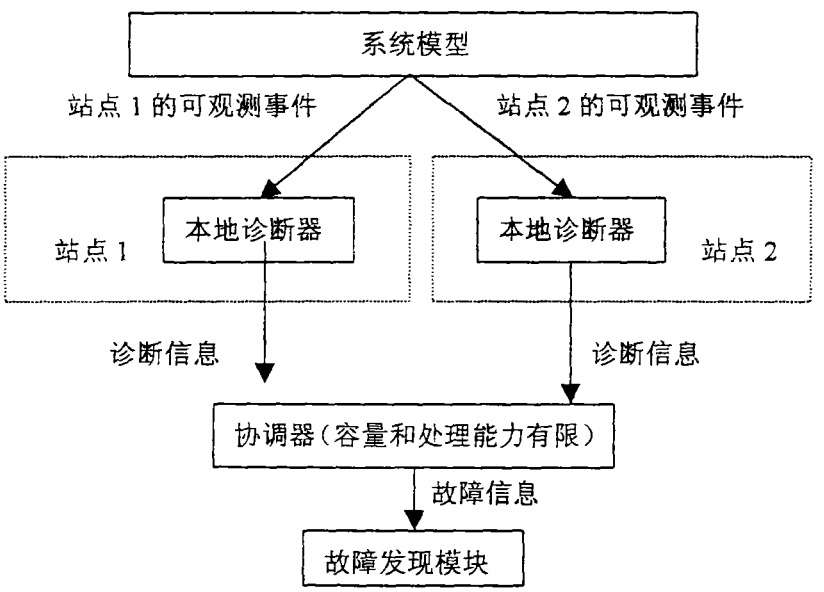


图 1.3 协调分布式的诊断体系结构原理图

继 Holloway 和 Chand 提出了基于时序(Time Template)的分布式监控方法[46]后，Debouk 等人在 99 年提出了分布式故障诊断的解决方案[61, 62]。这种方案将诊断器理论加以发展，使之适应分布式的情况。Debouk 等人建立了一个协调分布式的诊断体系结构，包含一系列本地站点、协调器以及故障发现模块。体系结构如图 1.3 所示。本地站点将各自的诊断信息交给协调器，协调器根据这些

信息和某些决策规则来判定故障发生情况，最后将故障信息交给故障发现模块。Debouk 还提出了三种协议来实现这种体系的诊断功能。这些协议在性能等方面有所不同，但每个协议都由本地站点产生的诊断信息、使用的通讯规则以及协调器的决策规则三个部分组成。Debouk 还扩展了可诊断性的概念，提出了在协议下的可诊断性。这种方案在离线时可判定给定系统在协议下的可诊断性。在该方案中，本地站点的诊断器与 Sampath 提出的诊断器相同或只稍微加以改变，真正实现了这种方法的分布式特性。在满足一定假设条件的情况下，该方案不但运行性能和集中式的诊断器相当，而且对协调分布式体系中存在的性能和复杂性的矛盾进行了很好的折中。

另一类分布式系统可称之为活跃系统（Active System）。这类系统可以看成是由许多可以互相通信的模块组成的网络，每个模块的行为可以用 DEDS 模型来表示。活跃系统平时处于静止状态，但是当接收到外部世界产生的世界后，则会产生响应；在响应过程中，系统将发生状态的变化，并产生一系列事件，同时也将接收外部世界的世界；当响应结束后，系统又恢复静止状态。Baroni 等人针对这类系统提出了一种基于模块的故障诊断方法[66, 67, 68]，相应的诊断方法以利用可观测的信息在线累进重构系统行为，从而产生诊断信息。该方法分三个步骤：（1）重构计划，即将行为重构问题多层次分解；（2）行为重构，即生成与每一组观测对应的所有动态行为；（3）诊断生成，根据重构后的系统模型及相应观测产生各个层次上的诊断信息，供诊断时使用。这种方法很好实现了分析复杂系统所必须解决的分解—合成的问题，有利于并行处理，从而使大规模活跃系统的诊断更有效，而以往一次生成全局模型的方法在现实中几乎是不可能的。

还有一类分布式系统则具有下面两个方面特征：

（1） 系统由许多组件构成，且存在共享资源。系统的各个组件由于需要访问共享资源，所以会与其他组件发生相互作用。

（2） 组件的数量和位置不是一成不变的，增加和减少组件比较频繁。

很明显，计算机和通信网络，制造业，过程控制，远程控制多机器人系统和电力系统等大规模分布式系统就具备上述特征。

由于上述特征，如果我们采取传统的集中式诊断方法，就得重构整个系统行为，并构造整个系统的诊断器。由于这类系统比较庞大和复杂，整个构造过程费

时费力；而且每当加入或者去除一个组件时，就得重新构建一切，而且这种改变是经常的，这极大地增加了开销。为了解决这个问题，我们可以针对每个组件建立一个本地诊断器，而不用考虑整个系统行为。这样当增加或去除一个组件的时候，和它无相互作用的组件的本地诊断器就不受任何影响。这种模块化诊断的思想便能较好的解决上面提出的问题。当然，采用模块化诊断的前提是没有属于整个系统范围的故障需要诊断。

Ricker 和 Fabre 采取了上述的模块化诊断思想来解决这类系统的诊断问题[69]，但他们采用的方法有着下面的前提条件：当系统中发生故障时，会产生可见的告警，且一个告警对应着几个不同的故障的发生。但在现实中，故障的发生不一定存在可见的告警，所以该方法在应用中存在着不足。

有一类系统必须考虑其中的时间因素.而上述故障诊断方法不考虑事件发生的时刻。为了解决这一问题， Zad 等人提出了赋时离散事件系统的故障诊断方法[71]。这种方法是以赋时离散事件系统的监控理论[72]作为基础的。在这种方法中，先将赋时自动机中的时间因素按特定时钟节拍转换成事件，从而把赋时自动机转化为无时序自动机，然后再采用基于状态的故障诊断方法对系统进行诊断。这样就很好的解决了这类系统的诊断问题。

以上的故障诊断方法都采用了自动机模型，也有很多学者采用 DEDS 的 Petri 网模型来研究故障诊断问题[65, 80]，比如 Aghasaryan 等提出的基于部分随机 Petri 网模型方法[65]。这类方法的主要思想是利用事件并发性来分离和简化有故障系统中的状态估计问题。它采用了 Petri 网及其因果语义来对并发性进行建模。Aghasaryan 还提出了部分随机 Petri 网模型来处理问题，从而在并发与独立运行间建立了某些等价关系。由于每类故障都对应于一个可见的警报，诊断问题被描述成在给定一个可观警报序列的前提下计算出网络的最可能的历史记录。

由于离散事件系统的深入研究和现代工业过程控制的需要，混杂系统的概念应运而生[13, 16]，它被定义为离散时间系统与连续变量系统相混合而形成的统一的动态系统。相应的，针对混杂系统的 DEDS 故障诊断方法也得到了发展[56, 59, 90]。

除了上面所提到的用于故障诊断的离散事件模型以外，还有一些学者也提出了一些以其他离散事件模型为基础的故障诊断方法，详细情况可见文献【51，55】。

1.2 本文研究的意义

在现代工业大生产和自动化生产领域中，生产过程涉及到计划调度、车间管理、单元控制、产品检验和过程维护等诸多环节，故障现象难以避免。为保证生产正常、有效地进行，必须对生产过程进行监测，对故障事件及时进行诊断和处理。对连续信号形式的故障现象，国内外学者已进行了大量卓有成效的工作，并在实际应用中取得了显著成果。但在实际的工业生产过程中，大量的故障却具有离散化和事件驱动的特征，如阀门的开关、电机的启停，位置的定位，多点液位控制等，系统状态并不具有连续性，传统的故障诊断理论难以进行有效的处理。因此，研究一种新型的、针对离散化故障事件的理论方法和体系，是十分必要的。

从离散事件动态系统概念的提出至今不过十多年时间，但在现代高新技术的推动下获得了迅速发展，出现了各种模型方法，成为继非线性理论、鲁棒控制理论之后研究复杂系统的又一热点，其中 DEDS 监控理论则是目前逻辑层次上研究 DEDS 的一个主流方法。近年来，DEDS 监控理论的理论体系日臻完美，研究深度日渐深入，应用范围逐步拓宽，由 CIMS、FSM、C^3I、机器人、交通控制、通讯阻塞、协议转换等领域扩展到工业生产的初步应用。目前，国外对 DEDS 监控理论的研究有两个趋势值得关注，一是继续深化和完善该理论体系，并在能观性、模块化控制及非确定性方面取得了一些突破性成果，另一方面进一步开拓该理论的应用领域，尝试将该理论成果应用于生产过程和故障检测及诊断技术中。由于计算机硬件技术的迅猛发展，使事件的并发处理、在线监控成为可能，从而为实际应用提供了技术保障。同时，由于 DEDS 监控理论已经日趋成熟，而现有的基于 CVDS 的故障诊断理论和技术又无法有效解决离散型事件故障问题，因此近来国外关于 DEDS 监控理论对该类故障建模和分析的文献日渐增多。

在我国，由于制造技术和工艺水平相对落后，各类设备、部件的故障发生率相对偏高，利用软技术解决不失为实现低成本自动化的有效途径。而基于 CVDS 的故障诊断理论和技术仅消除了连续性变量或工艺参数扰动给正常生产带来的不良影响，而大量离散型故障事件、突发故障仍为人工和经验处理。本项研究基于 DEDS 监控理论，试图建立离散型故障的模型，利用自动机强大的推理和语言

处理能力，解决该类故障诊断、控制问题，并进一步结合 CVDS 的故障诊断理论的成果，全面提高系统的故障诊断能力，不仅理论上有突破性意义，如应用于大型工业过程和自动化生产线，将产生显著的社会和经济效益。

基于离散事件动态系统监控理论的故障诊断技术是动态系统故障技术的一种新途径。与传统的故障诊断技术相比，尤其在分布式动态系统故障诊断情况下，至少具有如下优势：(1) 由于信号无衰减，诊断更可靠；(2) 由于输入、输出量均是数字量，实施更方便；(3) 对于分布式系统，故障诊断器和监控器的设计以及它们的集成将更方便。正是由于上述原因，尽管该理论发展历史不长，但其应用范围已渗透到计算机和通信网络、电力系统、集成电路制造、柔性生产线、间歇生产过程、多机器人的工作单元和高速公路的智能调度等领域。

1.3 全文概貌

本文主要研究以离散事件动态系统的监控理论为基础的诊断器理论和方法，同时也结合其他 DEDS 故障诊断方法，主要完成下面几个工作：

（1） 在第一章中全面、系统的阐述了 DEDS 故障诊断理论的产生原因和发展过程，总结了目前 DEDS 故障诊断的各种常见方法，简要的介绍了这些方法的基本原理，并对 DEDS 故障诊断的优点和意义进行了阐述。

（2） 本文中重点讨论的 DEDS 故障诊断方法是来源于监控理论，所以在第二章中我们较为详细地讲述了监控理论。我们首先介绍了 DEDS 监控理论中的一些基本知识和在监控理论中起比较重要作用的一些子语言及其优化语言的求解方法。这些都是后面章节讨论的基础。考虑到针对复杂系统的监控是监控理论的重要研究方向，所以本章也着重讲述了分散监控方法，包括模块化监控方法以及由此引出的针对局部目标和全局目标的分散监控方法。在前面论述的基础上，作者提出了用于全局目标分散监控的联合正态理论，该理论能较好的解决分散监控的实现方面的问题，而且对 DEDS 故障诊断理论研究也有一定的作用，比如在后面部分提出的分散活跃诊断方法就是以联合正态理论为基础的。

（3） 诊断器理论是 DEDS 诊断理论中的一个重要分支，本论文的研究主要是围绕诊断器理论进行的。所以在第三章中比较详细的介绍了诊断器理论，包

括概念，模型，诊断原理等。而针对不可诊断系统的活跃诊断方法则使得诊断器理论能实际中能更好的应用，所以本章也对活跃诊断方法进行了叙述。而这些都是基础性知识，我们将在本章的基础上提出了一些新的方法，这些将在后面章节进行介绍。

（4） 同监控理论一样，处理复杂系统一直是 DEDS 故障诊断理论的一个努力方向，而很多复杂系统都具有信息分散的特点，所以对信息分散系统的诊断就引起了广泛的关注。Debouk 提出的协调分布式故障诊断方法就很有代表性，所以在第四章中对它进行了介绍。在协调分布式方法和活跃诊断方法的基础上，我们提出了一种针对信息分散系统的分散式活跃诊断方法，使得诊断器理论能更好的应用于具有信息分散特征的系统，尤其是那些不具备可诊断性的系统。

（5） 像计算机网络之类的由相互作用的组件构成的大型分布式系统，进行集中式诊断代价太大。基于已有的一些采用模块化诊断来处理这类问题的方法，第五章提出了一种基于诊断器理论并运用模块化诊断的方法。该方法能有效处理结构变化比较大的复杂分布式系统的诊断问题，具有很好的现实意义和研究前景。

（6） DEDS 故障诊断方法不仅在理论研究上取得了较大的发展，而且已经开始广泛运用于实际的生产和生活中，所以第六章讨论了 DEDS 故障诊断方法的应用，并举了一些应用方面的例子。

在论文的最后，作者首先对全文进行总结，并讨论了 DEDS 故障诊断领域特别是监控器理论这方面所存在的一些悬而未决的问题，最后对 DEDS 故障诊断领域的研究和应用的前景进行了讨论。

第二章 DEDS 监控理论

2.1 引言

由于本文中将要介绍的 DEDS 的故障诊断理论都以 DEDS 监控理论为基础，所以有必要对 DEDS 监控理论做一个较为详细的介绍。

与传统的连续变量动态系统不同，离散事件动态系统是一种人造复杂系统，它不能用连续变量系统中的微分方程和差分方程进行有效的描述，因为它在时间或者状态空间上具有离散性。系统状态的变化不是因为实时时钟而是因为事件驱动，而且具有典型的非确定性。制造系统、交通系统、电力系统、操作系统、通信系统、信息系统以及工业过程广泛存在着上述特点，即具有离散性、瞬时性、并发性、异步性。

自 80 年代初 Y. C. Ho 正式提出离散事件动态系统以来，针对 DEDS 的研究非常活跃。研究离散事件动态系统所采用的模型有很多种，根据时间和非时间的特点，在逻辑层次、代数层次、性能层次上进行如表 2.1 的划分[12]。各种建模方法的介绍详见文献【16】。

	时间模型	非时间模型
逻辑层次	（时序逻辑）↔ （时间的）↔	有限自动机 Petri 网络
代数层次	极大极小代数	自回归过程 通讯序贯过程
性能层次	马尔柯夫链 排队网络 广义半马式过程	注：此项为随机型模型

表 2.1 DEDS 模型分类

DEDS 监控理论是 Wonham 和 Ramage[1] 于八十年代创立的，该理论用自动机来建模 DEDS，所以是属于逻辑层次的。与其他研究 DEDS 的方法比较，监控理论具有两个显著特点：（1）第一次把控制的思想应用到 DEDS 的研究中；（2）

开环和闭环可分开研究，克服了系统分析的复杂性。由于具有这些特点，针对监控理论的研究得到了很大的发展[1]~[16]，并扩展到复杂系统监控方面，比如模块控制和分散控制等[17]~[39]。此外，对监控理论应用方面的研究也取得了一系列进展[40]~[43]。

监控理论开始是以集中式系统为研究对象的，与之相对应的是集中式监控方法。但是随着系统规模的扩大和系统复杂性的增加，系统的状态数目急剧增加，采用集中式监控的方法不仅增加了分析上的难度，而且无法进行监控实施。而分散监控就是针对复杂系统提出的，其思想是通过对未被控系统行为或控制目标行为的分解，对各局部系统分别实施监控，从而达到全局监控的目的。分散监控可以广泛运用于各类信息分散系统或者分布式系统中，比如通讯网络、制造系统、交通系统等。

本章首先给出了 DEDS 监控理论的一些基本知识，然后对分散监控问题进行了讨论。最后，根据这些内容，作者提出一种联合正态理论，该理论用于全局目标的分散监控，能较好解决分散监控的实现方面的问题。

2.2 监控理论基础

2.2.1 建模方法

在监控理论中，一个离散事件系统可以用一个五元的自动机模型（也叫语言发生器）来表示：$G=(Q,\Sigma,\delta,q_0,Q_m)$。式中 Q 是状态 q 的集合，$q_0 \in Q$ 为初始状态，$Q_m \subseteq Q$ 称为标记（或终止）状态子集，Σ 是输出符号 σ 的集合，又称为符号表。$\delta:\Sigma\times Q\to Q$ 为状态迁移函数，它是一个部分函数。G 可以看成是一个有向图，其节点集为 Q；每一个使 $q'=\delta(\sigma,q)$ 成立的三元组（σ,q,q'）可用一条标上 σ 的 $q\to q'$ 的边来表示，这样的边（即状态迁移）称为一个事件。当然，可以简单的用 σ 来表示一个事件。如此，可以把自动机看成一个装置：初始状态为 q_0，通过执行状态转移产生事件序列（即由 Σ 中事件组成的字符串），而事件的产生被认为是自发的、异步的和瞬时的。这样系统的行为就可以用事件序列或状态序列来表征。

如果用Σ^*表示Σ中所有有限长字符串的集合（包括空串ε），就可以按照下面定义将状态迁移函数的进行扩展，$\delta:\Sigma^*\times Q\to Q$：

$$\begin{cases}\delta(\varepsilon,q)=q, q\in Q\\ \delta(s\sigma,q)=\delta(\sigma,\delta(s,q)), \text{当}\delta(s,q)!\text{且}\delta(s,q')!\text{且}q'=\delta(s,q)\end{cases}$$

于是，我们可以定义一个在Σ上的由G产生的语言：

$L(G)\triangleq\{\omega:\omega\in\Sigma^*\text{且}\delta(\omega,q_0)!\}$

而由G标识的语言：$L(G)\triangleq\{\omega:\omega\in L(G)\text{且}\delta(\omega,q_0)\in Q_m\}$。明显的，$L_m(G)\subseteq L(G)\subseteq\Sigma^*$。

下面用泵来举例说明离散事件动态系统的建模。泵有两种状态：泵打开（PON）、泵关闭（POF）；两个事件：打开泵（START_PUMP）、关闭泵（CLOSE_PUMP）。假设泵开始的时候是关的，最后可以是开的，也可以是闭的。即Q={PON,POF}，Σ={START_PUMP，CLOSE_PUMP}，q_0=POFF，$Q_m=Q$。其状态迁移函数可以用下面的自动机来表示：

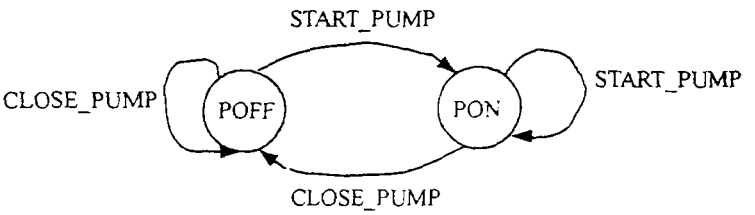


图2.1 一个简单的离散事件动态系统的例子

我们可以定义G中的能达部分，以消去从q_0出发未达到的状态。定义如下：

$A_c(G)\triangleq(Q_{ac},\Sigma,\delta_{ac},q_0,Q_{ac,m})$，其中：$Q_{ac}\triangleq\{q:\exists\omega\in\Sigma^*,\delta(\omega,q_0)=q\}$，

$Q_{ac,m}\triangleq Q_{ac}\cap Q_m$，$\delta_{ac}\triangleq\delta|(\Sigma\times Q_{ac})$

如果$G=A_c(G)$，则称G是能达的(accessible)。如果L(G)中每一个字符串均可完成在$L_m(G)$中，即$(\forall\omega)\omega\in L(G)\Rightarrow(\exists s)s\in\Sigma^*\text{且}\omega s\in L_m(G)$，则称G是协能达（co-accessible）。若G既是能达的又是协能达的，则称G为trim发生器。对于任意一个包含于Σ^*的子集，对应的trim发生器是唯一的。

定义 2.1：令 $L \subseteq \Sigma^*$，L 的闭集 $\bar{L}$ 表示 L 中所以字符串前缀的集合，即 $\bar{L} = \{s : s \in \Sigma^* 且 (\exists t) t \in \Sigma^* 使 st \in L\}$。当 $L = \bar{L}$ 时，称 L 为闭语言。

由该定义知，L(G)是闭语言，且 G 是 trim 的时候，$L(G) = \bar{L}_m(G)$。

基于自动机的建模方法使得系统设计问题可以归纳为一个逻辑设计问题，目的是使之出现期望的“好”的串而不是不期望的“坏”的串。这也使得在自动机模型框架下，对 DEDS 的研究集中于系统的控制问题，涉及逻辑层次下的控制特性、综合方法和控制器简化等问题。

2.2.2 能控性和能观性

如果一个事件是可以被禁止发生的，则称它为可控的；否则就称为不可控的。可控事件的集合 $\Sigma_c \subseteq \Sigma$ 称为可控事件集；不可控事件的集合 $\Sigma_{uc} \subseteq \Sigma$ 称为不可控事件集。在制造系统中，“工件到达”这个事件就是可控的；而“机床故障”这类事件就是不可控的。相应的，如果一个事件的发生可以由观测者（控制器）观测到，则称为可观事件；否则就称为不可观测事件。同样的可以定义可观测事件集 $\Sigma_o \subseteq \Sigma$ 和不可观测事件集 $\Sigma_{uo} \subseteq \Sigma$。显而易见，$\Sigma = \Sigma_c \cup \Sigma_{uc} = \Sigma_o \cup \Sigma_{uo}$，$\Sigma_c \cap \Sigma_{uc} = \Sigma_o \cap \Sigma_{uo} = \phi$。

能控性和能观性是设计控制器的最基本概念。下面分别进行介绍：

1、能控性

定义 2.2：令语言 $K \subseteq \Sigma^*, L \subseteq \Sigma^*$.

（1）若 $K = \bar{K} \cap L$，称 K 是 L 闭；（2）$\bar{K}\Sigma_{uc} \cap L \subseteq \bar{K}$，称 K 是（$\Sigma_{uc}$, L）不变；（3）$K \subseteq L(G)$，且 K 是（$\Sigma_{uc}$, L(G)）不变，则称 K 关于 L(G)能控。

语言 K 的能控性可以作如下解释：把 L(G)看成物理上可能发生的行为的集合，而 K 是合法的允许行为的集合；那么任一字符串 $s\sigma \in L(G)$ $(s \in \bar{K}, \sigma \in \Sigma_{uc})$ 都相当于一个合法行为。这就是说，在任意合法行为后面加上一个不可控事件组成新行为，如果该新行为在物理上是可能的，则它仍然是合法的容许行为。根据能控性定义，可以得到如下的性质：

定理 2.1：

（1）K 能控当且仅当 $\bar{K}$ 能控；

（2）若 K_1、K_2 能控，则有 $K_1 \cup K_2$ 能控，而且如果 K_1、K_2 非冲突（nonflicting，即 $\overline{K_1 \cap K_2} = \bar{K}_1 \cap \bar{K}_2$）或闭，则 $K_1 \cap K_2$ 能控。

（3）若 K 关于 M 能控，M 关于 L 能控，则 K 关于 L 能控。

（4）若 K 关于 L 能控，且 $M \subseteq L$，则 K 关于 M 能控。

2、能观性

根据事件集划分 $\Sigma = \Sigma_o \cup \Sigma_{uo}$，可以引入映射 P：$\Sigma^* \to \Sigma_o^*$，定义如下：

$$\begin{cases} P\varepsilon = \varepsilon; P\sigma = \varepsilon, \sigma \in \Sigma_{uo}; P\sigma = \sigma, \sigma \in \Sigma_o \\ P(s\sigma) = P(s)P(\sigma), s \in \Sigma^*, \sigma \in \Sigma \end{cases}$$

若 $s, s' \in \Sigma^*$ 且 $P(s) = P(s')$，则称 $(s, s') \in KerP$，记为 $s \equiv s' \pmod{KerP}$。显然 KerP 是个右同余关系。

定义 2.3：称 K 关于 $L_m(G)$ 能观，若对 $s, s' \in \Sigma^*$，$(s, s') \in$ KerP，满足：

（1）$(s, s') \in nextact_K$，即 $(\forall\sigma) s\sigma \in \bar{K} \wedge s' \in \bar{K} \wedge s'\sigma \in L(G) \Rightarrow s'\sigma \in \bar{K}$；

（2）$(s, s') \in markact_K$，即 $s \in K \wedge s' \in \bar{K} \cap L_m(G) \Rightarrow s' \in K$；

（3）交换（1）、（2）中的 s、s′ 仍然成立。

在该定义中，当所讨论的语言不涉及 $L_m(G)$ 时（相当于 $Q_m = Q$），则条件（2）自动满足。

语言能观的概念也可以这样解释：从观测者来看，尽管 G 的输出 P(s)与 P(s′) 相同，但若 s 导致与 s′ 相同的下一行为 σ，且 s 导致与 s′ 相同的标记行为，则认为语言 K 能观。

定义 2.4：称语言 K 是关于(L(G), P)正态（normal），若 $K = L(G) \cap P^{-1}P(K)$。

定理 2.2 [4]：若 K 是闭或 $L_m(G)$ 闭，$K \subseteq L(G)$，则当 K 是正态的时候，K 能观。

正态性在监控器设计中经常代替能观性，因为正态性便于判断且求正态子语言比较容易。下面给出能观性的一些性质：

定理 2.3：[4]

（1）K 能观当且仅当 $\bar{K}$ 能观；

（2）若K_1、K_2能观，如果K_1、K_2非冲突或闭，则$K_1 \cap K_2$能观。

（3）若 K 关于 M 能观，M 关于 L 能观，则 K 关于 L 能观。

（4）若 K 关于 L 能观，且$M \subseteq L$，则 K 关于 M 能观。

2.2.3 监控器

2.2.3.1 监控器基础

所谓监控问题，就是在各种约束条件下，寻找并构造适当的监控器，使得闭环系统的行为满足特定的控制要求。监控器的设计是监控理论的核心，其基本思想就是使被控系统的行为等于期望行为 K 或处于合法行为 A 和最小容许行为 E 之间。在自动机模型框架下，系统的行为可以看成一被逻辑时间隔离开的字符串$s = \sigma_1\sigma_2\cdots\sigma_n$的集合，而要求系统遵循的规则或者约束也可以用事件串的集合来表示。这样，监控器的设计实质上就是在逻辑层次上对形式语言进行求解。

为了对系统 G 实施控制，引入映射$\Gamma = \{0,1\}^{\Sigma_c}$,即对任意$\gamma \in \Gamma$，有如下定义：

$$\gamma(\sigma) = \begin{cases} \{0,1\}, \sigma \in \Sigma_c \\ 1, \sigma \in \Sigma_{uc} \end{cases}$$

由此可见，Γ为Σ到$\{0,1\}$的一些映射的集合。当$\gamma(\sigma)$=1 时，允许事件σ发生；如果$\gamma(\sigma)=0$，则禁止σ发生。对所有$\sigma \in \Sigma_{uc}$，$\gamma(\sigma)$总是 1，因为我们无法禁止不可控事件的发生。

于是，我们根据 G 可以定义一个被控的离散事件过程：$G_c = (Q, \Gamma \times \Sigma, \delta_c, q_0, Q_m)$，其中$\delta_c : \Gamma \times \Sigma \times Q \to Q$，定义如下：

$$\delta_c(\gamma, \sigma, q) \triangleq \begin{cases} \delta(\sigma, q)，若\ \delta(\sigma, q)!\ 且\ \gamma(\sigma) = 1 \\ 无定义，否则 \end{cases}$$

为了获得一个理想的系统行为，需要有一个实施控制的装置，即监控器（Supervisor）。一个监控器在形式上可以定义成：$\Psi \triangleq (S, \Phi)$。$S = (X, \Sigma, \xi, x_0, X_m)$是一个确定性自动机，其中部分迁移函数$\xi : \Sigma \times X \to X$。而$\Phi : X \to \Gamma$则是一个把监控器状态 x 映射到控制模式$\gamma$上的全函数，对任意$x \in X$，$\Phi(x) = \gamma$且$\gamma \in \Gamma$。因此我们可以得到闭环形式的被监控的离散事件过程（SDEP）：

$$\Psi/G \triangleq A_c(X\times Q,\Sigma,\xi\times\delta_c,(x_0,q_0),X_m\times Q_m)。$$

其中部分函数$\xi\times\delta_c:\Sigma\times X\times Q\to X\times Q$，定义如下：

$$\xi\times\delta_c(\sigma,x,q)\triangleq\begin{cases}(\xi(\sigma,q),\delta(\sigma,q))，当\delta(\sigma,q)!、\xi(\sigma,x)!且\Phi(x)(\sigma)=1\\ 无定义，否则\end{cases}$$

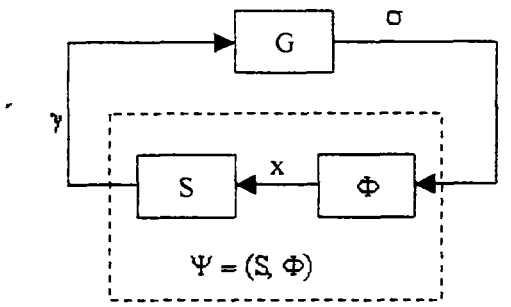


图 2.2 监控系统原理图

由监控器的定义可知，S 是观测器或者动态补偿器，而Φ是反馈部分。这样被控系统和监控器就组成了一个闭环系统，如图 2.2 所示,其中$\sigma\in\Sigma,\gamma\in\Gamma,x\in X$。

下面给出监控器的一些相关定义：

定义 2.5：称Ψ关于G_c是完备的（complete），如果$\forall s\in\Sigma^*,\sigma\in\Sigma$，满足：

$s\in L(\Psi/G)\wedge s\sigma\in L(G)\wedge\Phi(\xi(s,x_0))(\sigma)=1\Rightarrow s\sigma\in L(\Psi/G)$。

一般总假设Ψ关于G_c是完备的，否则只需对 S 作适当修改便能满足这个条件。与 L(G)和$L_m(G)$定义相似，我们可以定义$L(\Psi/G)$和$L_m(\Psi/G)$。再定义 G 中被Ψ控制的语言：$L_C(\Psi/G)\overset{\Delta}{=}L(\Psi/G)\wedge L_m(G)$。根据上述定义，有：$L_m(\Psi/G)\subseteq L_C(\Psi/G)\subseteq L_m(G)$。而当 G 是 trim 时，$\bar{L}_m(\Psi/G)\subseteq\bar{L}_C(\Psi/G)\subseteq L(\Psi/G)\subseteq L(G)=\bar{L}_m(G)$。

定义 2.6：

（1）称监控器Ψ关于 G 是非阻塞的（nonblocking），当$\bar{L}_c(\Psi/G)=L(\Psi/G)$。

（2）称监控器Ψ关于 G 是非排斥的（nonrejecting），当$\bar{L}_c(\Psi/G)=L_m(\Psi/G)$。

（3）称监控器Ψ关于 G 是正则的（proper），当Ψ关于 G 是完备、非阻塞和非

排斥时。此时，$\bar{L}_m(\Psi/G)=\bar{L}_c(\Psi/G)=L(\Psi/G)$。

2.2.3.2 监控器综合

监控器综合就是设计一个合适的监控器，通过对输入事件（即被控对象 G 的输出，见图 2.1）进行“允许”和“禁止”，使整个闭环系统的行为达到期望要求。因为监控器综合问题就是讨论监控器存在条件以及如何设计监控器这两个方面。

下面给出两个相关的语言类：

$$C(L)\triangleq\{K|K\subseteq L\wedge\bar{K}\Sigma_{uc}\cap L(G)\subseteq\bar{K}\},$$

$$F(L)\triangleq\{K|K\subseteq L\wedge K=\bar{K}\cap L_m(G)\}$$

根据定理 2.1，supC(L)和 supF(L)都是存在的。下面给出系统完全能观情况下监控器的存在条件：

定理 2.4[5]：令$\phi\neq K\subseteq L_m(G)$：

（1） 当且仅当 K 能控时，存在一个正则监控器 Ψ 使 $L_m(\Psi/G)=K$，$L_c(\Psi/G)=L_m(G)\cap\bar{K}$，$L(\Psi/G)=\bar{K}$；

（2） 当且仅当 K 能控且为 $L_m(G)$ 闭时，存在一个正则监控器 Ψ 使 $L_c(\Psi/G)=K$。

在定理 2.4 中，（1）讨论了监控标识（SMP）问题，（2）讨论了监控控制问题（SCP）。如果期望语言 K 不能满足能控性，可以用 K 的最大能控子集 $K^{\uparrow}$（即 supC(K)）来逼近。当期望行为是一个范围而非一个符合定理 2.4 中条件的语言时，即要求设计一个监控器满足：$E\subseteq L_m(\Psi/G)\subseteq A$ 或 $E\subseteq L_C(\Psi/G)\subseteq A$。根据定理 2.4 可知，当且仅当 $supC(A)\supseteq E$ 或者 $sup(C(A)\cap F(A))\supseteq E$ 时，存在监控器 Ψ，满足 $L(\Psi/G)=supC(A)$ 或者 $L(\Psi/G)=sup(C(A)\cap F(A))$。其中，A 一般表示“合法行为”，而 E 表示“最小的能接受行为”。

上面的讨论都是基于监控器能完全观测系统中所有事件的情况。如果系统事件集只是部分能观时，定义下面两个语言类：

$\underline{C}(L) \triangleq h\{K : K \subseteq L$ 且 K 是关于 L(G)的能控闭语言$\}$

$\underline{O}(L) \triangleq \{K : K \supseteq L$ 且 K 是关于 L(G)的能观闭语言$\}$

同样的，可以得出如下结论[3, 4]：

定理 2.5：

（1）$\phi \neq K \subseteq L_m(G)$，当且仅当 K 完全能控和完全能观时，存在监控器 Ψ 满足 $L_m(\Psi/G) = K$；

（2）$\phi \neq A \subseteq E \subseteq L(G)$，当且仅当 $\inf \underline{O}(A) \subseteq \sup \underline{C}(E)$ 时，存在监控器 Ψ 满足 $A \subseteq L(\Psi/G) \subseteq E$。

定理 2.5 的（2）可以发展成为 SCPO[5, 12, 16]（Supervisory Control Problem with Observation）问题，即要求设计监控器满足：$\bar{A} \subseteq L(\Psi/G) \subseteq \bar{E}$，详情可见附录 A。

由于系统行为可以用语言表示，所以求解一定条件下的子语言在监控器理论中有重要的意义。附录 B 比较系统的介绍了在监控理论中起重要作用的各类语言及其子语言的求解方法，以供参考。

2.3 分散监控

本节主要介绍分散监控中常见的两种不同的控制策略，而这两种策略都是以模块化方法为基础的，关于模块化方法的内容可见附录 C。

2.3.1 基于局部控制目标的分散监控

基于局部控制目标的分散监控问题是指，对给定的被控系统的自动机 G 及其产生的语言 L(G)，将指定的一个目标语言 $L \subset L(G)$ 合理地分解成若干个子任务，并采用相应个数的局部监控器去分别控制每个子任务，同时保证总的控制性能仍保持为目标语言 L 的最大能控子语言即 supC(L)。其基本思路是，通过对目标语言的分解来实现把监控器构造成多个局部监控器，从而使每个监控器的结构在复杂性上得到很大的简化。这种方法可以应用于分布式系统中[32]~[37]。

令 $G = (Q, \Sigma, \delta, q_0, Q_m)$，由于每个子任务都可以看成一个子系统，设

$G_i=(Q_i,\Sigma_i,\delta_i,q_{0i},Q_{mi})$ 为其中一个子系统，各子系统的组合 $\oplus G_i=G$。其中，

$\Sigma=\bigcup_i\Sigma_i$，$\Sigma_i\cap\Sigma_j\neq\phi$，$i\neq j$。$\Sigma_c=\bigcup_i\Sigma_{ic}$，$\Sigma_{uc}=\bigcup_i\Sigma_{iuc}$，

$Q=Ac(Q_1\times Q_2\times\cdots\times Q_n)$，$Q_m=Ac(Q_{m1}\times Q_{m2}\times\cdots\times Q_{mn})$，

$q_0=(q_{01},q_{02},\cdots,q_{0n})$，

定义 $I(\sigma)\triangleq\{i|\sigma\in\Sigma_i\}$，则状态迁移函数 $\delta:\Sigma\times Q\to Q$，定义如下：

$$\delta(\sigma,(q_1,q_2\cdots q_n))=\begin{cases}(q_1',q_2',\cdots,q_n'),\ \text{若}\forall i\in I(\sigma),\ \delta_i(\sigma,q_i)!\\ \text{无定义，否则}\end{cases}$$

$$q_i'=\begin{cases}\delta_i(\sigma,q_i),\ \text{当}i\in I(\sigma)\\ q_i,\ \text{当}i\notin I(\sigma)\end{cases}$$

其中 $q_i\in Q_i$，

根据上面的定义，G 称之为 $\{G_i\}$ 的织积[32]（shuffle product），记为 $\|$。则分别有 $G=\|_{i=1}^{n}G_i$ 和 $L(G)=\|_{i=1}^{n}L(G_i)$。

定义全局事件集合到局部事件集合的映射 $\pi_i:\Sigma^*\to\Sigma_i^*$，

$\pi_i(\varepsilon)=\varepsilon$，$\pi_i(\sigma)=\begin{cases}\sigma,\ \text{若}\sigma\in\Sigma_i\\ \varepsilon,\ \text{否则}\end{cases}$，$\pi_i(s\sigma)=\pi_i(s)\pi_i(\sigma)$。

定理 **2.6**[12]：

（1）$L(G)=\bigcap_{i=1}^{n}\{s\in\Sigma^*:\pi_i(s)\in L(G_i)\}=\bigcap_{i=1}^{n}\pi_i^{-1}L(G_i)=\|_{i=1}^{n}L(G_i)$；

（2）设 $L_i\subseteq L(G_i)$，$1\le i\le n$，且 L_i 闭，对 $L(G_i)$ 能控，若令 $L=\|_{i=1}^{n}L_i$，则 $L\subseteq L(G)$，且 L 闭，对 L(G)能控。

该定理说明，在满足（2）的条件下，分散监控和集中监控是等效的。

如果各局部监控器目标 L_i 不满足局部能控和 $L_m(G_i)$ 闭的要求，则可定义语言集合：$C(L_i)\triangleq\{K:K\subseteq L_i\wedge K=\bar{K}\cap L_m(G_i)\wedge\bar{K}\Sigma_{uc}\cap L(G_i)\subseteq\bar{K}\}$，$C(L_i)$ 存在唯一的最大子语言，记 $K'\triangleq\sup C(L_i)$。这时可以用 K' 来代替 L_i 来构造分散监控器。

2.3.2 基于全局目标的分散监控

在 2.3.1 中，我们是用每个局部监控器来监控每个子任务，而本节中的思路则是采用多个局部监控器来同时监控一个被控系统，以实现期望的控制性能。在现实中，由于受到地理位置或者其它因素的影响，经常难以对被控系统实施有效的集中监控，而基于全局目标的分散监控思想就很好的解决了这一问题。所以这种基于全局目标的分散监控方法常常应用于各类信息分散系统（比如计算机和通信网络，制造业，过程控制，远程控制多机器人系统和电力系统等）中[17]~[31]。系统模型如下：

给定一个自动机形式的系统 $G=(Q,\Sigma,\delta_c,q_0,Q_m)$：可控事件集合 Σ_c，可观事件集合 Σ_o，一个对事件集合 Σ 划分 I={1,2,...n}，局部事件集 Σ_i，局部可观事件集 Σ_{io}，局部可控事件集 $\Sigma_{ic}(i\in I)$；满足 $\Sigma=\bigcup_i\Sigma_i$，$\Sigma_c=\bigcup_i\Sigma_{ic}$，$\Sigma_o=\bigcup_i\Sigma_{io}$，而且各个划分可能有共同事件。设映射 P_i：$\Sigma\to\Sigma_{io}$，将其扩展为 P_i：$\Sigma^*\to\Sigma_{io}^*$，有：

$$P_i(\varepsilon)=\varepsilon,\quad P_i(s\sigma)=P_i(s)P_i(\sigma),\quad P_i(\sigma)\begin{cases}\sigma, & \text{如果}\sigma\in\Sigma_{io}\\ \varepsilon, & \text{否则}\end{cases}$$

设各个局部监控器 $\psi_i=(S_i,\varphi_i)$ 为确定性自动机，$S_i=(X_i,\Sigma_{io},g_i,x_{i0},X_{im})$

$$\varphi_i(x_i)(\sigma)=\begin{cases}1, & \text{如果}\sigma\notin\Sigma_{ic}\\ \{0,1\}, & \text{如果}\sigma\in\Sigma_{ic}\end{cases}\qquad g_i(x_i,\sigma)=\begin{cases}x_i', & \text{如果}\sigma\in\Sigma_{io}\\ x_i, & \text{如果}\sigma\notin\Sigma_{io}\end{cases}$$

则 $\Psi_i/G=(X_i\times Q,\ \Sigma,\ (g_i\circ P_i)\times\delta_c,\ (x_{i0},q_0),\ \bar{X}_{im}\times Q)$.

$$(g_i\circ P_i)\times\delta_c(x_i,q)=\begin{cases}(x_i',\ q'),\ \text{如果}g_i(x_i,\sigma)=x_i',\ \Psi_i(x_i)(\sigma)=1,\ q'=\delta(q,\sigma)\\ (x_i,\ q'),\ \text{如果}g_i(x_i,\sigma)=x_i,\ \Psi_i(x_i)(\sigma)=1,\ q'=\delta(q,\sigma)\\ \text{无定义，否则}\end{cases}$$

用 $\Psi=(S,\varphi)$ 表示各局部监控器的集成，$S=(X,\Sigma_o,g,x_0,X_m)$，其中：

$$X=X_1\times X_2\times\cdots\times X_n,\ X=X_{1m}\times X_{2m}\times\cdots\times X_{nm},\ x_0=(x_{10},x_{20},\cdots,x_{n0}),$$

$$\Sigma_o=\prod_{i=1}^{n}\Sigma_{io},\ \varphi(x_1,x_2,\cdots,x_n)=\varphi_1(x_1)\wedge\varphi_2(x_2)\wedge\cdots\wedge\varphi_n(x_n),$$

$$g((\sigma_1,\sigma_2,\cdots,\sigma_n)(x_1,x_2,\cdots,x_n))=(g_1(x_1,\sigma_1),\ g_2(x_2,\sigma_2),\ \cdots,\ g_n(x_n,\sigma_n))$$

记 $(S,\varphi)=(\{S_i\},\{\varphi_i\})$，$P(s)=(P_1(s),P_2(s),\cdots,P_n(s))$。

类似的，可以定义分散监控器 Ψ 完备性：

定义 2.7：称 Ψ 关于 G 是完备的，如果 $\forall s\in\Sigma^*,\sigma\in\Sigma$，满足：

$$s\in L(\Psi/G)\wedge s\sigma\in L(G)\wedge\varphi[g(P(s)),x_0](\sigma)=1\Rightarrow s\sigma\in L(\Psi/G)$$

与集中监控中的能观性类似，下面给出分散监控中的联合能观的概念：

定义 2.8[3]：$\overline{K}\subseteq L(G)$ 是联合能观的，如果对 $\forall\sigma\in\Sigma$，K 中序列串 $\{s_i\}$，$i\in I(\sigma)$，$I(\sigma)=\{i:\sigma\in\Sigma_i\}$，$s'\in\overline{K}$，由：

(1) $s_i\sigma\in\overline{K}$, $\forall i\in I(\sigma)$;

(2) $s'\sigma\in L(G)$;

(3) $P_i(s_i)=P_i(s')$, $\forall i\in I(\sigma)$;

可得出 $s'\sigma\in\overline{K}$。

如果 $\Sigma_i\cap\Sigma_j=\phi$，$i\neq j$，联合能观性可通过验证每个站点的局部能观性得到。下面给出分散监控器存在的充分必要条件：

定理 2.7[3]：令 $K\subseteq L_m(G)$，$K\neq\Phi$，当且仅当：

（1）$K=\overline{K}\cap L_m(G)$；

（2）$\overline{K}\Sigma_{uc}\cap L(G)\subseteq\overline{K}$；

（3）K 是联合能观的。

则存在分散监控器 $S=(\{S_i\},\{\varphi_i\})$，使得 $L_c(S/G)=K, L(S/G)=\overline{K}$。

该定理表明，在 K 满足一定条件的情况下，分散监控可以和集中监控等效，而且从全局看，所得到的监控是非阻塞的。

在条件满足的情况下，可用如下的方法构造分散监控器：

（1）令 $R=(\Sigma, X, \delta, x_0, X_m)$ 为 K 的识别器，即 L(R)=K。当 K 正规时，X 有穷。

$$\xi_i(\sigma, x_i)=\begin{cases}\{\delta(s, x): s\in\Sigma^*, x\in x_i, P_i(s)=\sigma\}, & \text{若非空}\\ \text{无定义}, & \text{否则}\end{cases}$$

（2）定义分散监控器 (S_i, φ_i)，$S_i=(\Sigma_{ic}, X_i, \xi_i, x_{0i}, X_{mi})$，$X_i$ 为 X 的有穷状态子集，对 $\forall\sigma\in\Sigma_{ic}$，$x_i\in X_i$，有 $x_{0i}=\{\delta(s, x_0): s\in\Sigma^*, P_i(s)=\varepsilon\}$，

$X_{mi}=\{x_i\in X_i, (\exists x\in x_i)x\in X_m\}$。

（3） 由（2）可得 $L(S_i)=P_iK$，定义 φ_i 如下：

$$(\forall\sigma\in\Sigma_{ic},\ x_i\in X_i)\varphi_i(\sigma,\ x_i)=\begin{cases}1, & (\exists x\in x_i)\delta(\sigma,\ x)!\\ 0, & \text{否则}\end{cases}$$

2.4 用于分散监控的联合正态理论

针对复杂系统的监控问题提出的分散监控理论已经得到了广泛运用。分散监控器存在的充要条件是监控器作用产生的闭环被控语言是能控且联合能观的[3]、[18]、[20]、[30]。在分散监控的应用中，常常需要给定系统的合法行为，通过分散监控使系统的行为被限制在合法行为之内。而解决这类问题的关键就是求给定语言的能控和联合能观的子语言集或超语言集。很多文章已经给出了求超集下界的方法，而子集的上界已经被证明是无法求出的。Yoo 提出用强可分解性[30]来代替联合能观，并说明给定语言的能控及强分解子集上界存在，可是没有给出求解的形式，无法应用于实际。

本节则使用了联合正态理论来解决上面提出的问题。在该理论中，用联合正态的概念取代联合能观作为分散监控器存在的条件，并给出给定闭语言 K 的闭、能控及联合正态的一个子语言，即 isupCUN(K)。虽然该子语言不是上界，但可以证明其比能控及强分解性的上界要大。在具体应用中可以求出合法行为 K 的子集 isupCUN(K)，并将它作为系统采用分散监控后的闭环行为。这样，通过分散监控器的作用，系统的行为就被限制在合法行为之内。在下面的讨论中，所用系统模型与 2.3.2 中的相同。

2.4.1 联合正态

从 2.3.2 的讨论中可知，要实现对语言 K 的分散监控，K 必须是联合能观的。不过，验证 K 的联合能观性是比较复杂的。此外，如果 K 不满足条件，K 的 $L_m(G)$ 闭、能控、联合能观的子语言不存在唯一的最大元素，因为联合能观语言的并运算不封闭。总之，该定理的条件不容易检验，实用性不强。针对这个问题，我们给出一个联合正态的概念，来取代联合能观：

定义 **2.9**[20]：称 K 关于 $(L(G),\{P_i\})(i\in I)$ 联合正态，如果：

$$K=L(G)\cap\bigcap_i P_i^{-1}P_i(K)$$

从联合正态的定义可以推出下面的结论：

推论 2.1：K 关于 $(L(G), \{P_i\})(i \in I)$ 联合正态，当且仅当 $\forall s \in L(G)$，满足：

$$((\forall i \in I)(P_i(s) \in P_i(K))) \Leftrightarrow s \in K$$

推论 2.2：对任意语言 $K \subseteq L(G)$，$L(G) \cap \bigcap_i P_i^{-1}P_i(K)$ 关于 $(L(G), \{P_i\})$ 联合正态。

证明：

$(\Rightarrow)$: $\forall s \in L(G)$, 有

$$(\forall i \in I)(P_i(s) \in P_i(L(G) \cap \bigcap_i P_i^{-1}P_i(K))) \Rightarrow (\forall i \in I)(P_i(s) \in P_i(L(G) \cap P_i^{-1}P_i(K)))$$

$$\{L(G) \cap \bigcap_i P_i^{-1}P_i(K)) \subseteq L(G) \cap P_i^{-1}P_i(K) \Rightarrow P_i(L(G) \cap \bigcap_i P_i^{-1}P_i(K)) \subseteq P_i(L(G) \cap P_i^{-1}P_i(K))\}$$

$$\Rightarrow (\forall i \in I)(P_i(s) \in P_i(K)) \Rightarrow (\forall i \in I)(s \in P_i^{-1}P_i(K)) \Rightarrow s \in L(G) \cap \bigcap_i P_i^{-1}P_i(K))$$

$(\Leftarrow)$: 对任何语言集合都满足。

得证。

推论 2.3：如果 K 关于 $(L(G), \{P_i\})$ 联合正态，则 K 关于 $(L(G), P)$ 正态。

证明：因为 K 关于 $(L(G), \{P_i\})$ 联合正态，所以有 $K = L(G) \cap \bigcap_i P_i^{-1}P_i(K)$，而显然有 $(\forall i \in I)\, P_i^{-1}P_i(K) \supseteq P^{-1}P(K)$，所以 $K = L(G) \cap \bigcap_i P_i^{-1}P_i(K) \supseteq L(G) \cap P^{-1}P(K)$，由于 $K \subseteq L(G) \cap P^{-1}P(K)$，所以有 $K = L(G) \cap P^{-1}P(K)$，即 K 关于 $(L(G), P)$ 正态。

下面定理说明联合正态的约束要强于联合能观：

定理 2.8：如果 K 关于 $(L(G), \{P_i\})$ 联合正态且 K 闭，则 K 关于 $(L(G), \{P_i\})$ 联合能观。$(i \in I)$

证明：从联合能观的定义出发：

由于 $K = \overline{K}$ 则取 K 中序列串 $\{s_i\}$，$i \in I(\sigma)$，$I(\sigma) = \{i: \sigma \in \Sigma_i\}$，$s' \in \overline{K}$，取任意 $\sigma \in \Sigma$ 满足：

（1）$s_i\sigma \in K$, $\forall i \in I(\sigma)$；

（2）$s'\sigma \in L(G)$；

（3）$P_i(s_i) = P_i(s')$, $\forall i \in I(\sigma)$。

现在只要证明 $s'\sigma \in K$ 即可。对任意 $i \in I(\sigma)$，由（3）得 $P_i(s_i\sigma) = P_i(s'\sigma)$；由（1）得 $s'\sigma \in P_i^{-1}P_i(K)$；则 $s'\sigma \in \bigcap_i P_i^{-1}P_i(K)$，结合（2）得 $s'\sigma \in L(G) \cap \bigcap_i P_i^{-1}P_i(K)$。由于 K 是联合正态的，由定义 2.9 得：$s'\sigma \in K$。

得证。

由定理 2.7 和 2.8 可以得到下面的推论：

推论 2.4：令 $K \subseteq L(G), K \neq \Phi$，当满足下面三个条件时，存在一个分散监控器 $S = (\{S_i\}, \{\varphi_i\})(i \in I)$，使得 L(S/G)=K：

（1）$K = \overline{K}$；

（2）K 关于 $(L(G), \Sigma_{uc})$ 能控；

（3）K 关于 $(L(G), \{P_i\})$ 联合正态。

上面的推论说明，在讨论分散监控器存在时，可以用这个充分条件证明。且任何一个语言的能控、联合正态且闭子语言可以成为分散监控器构造的目标。

下面讨论 isupCUN(K)的一些性质。我们先给出一个强联合正态的概念（与强可分解性[30]（strongly decomposability）是同一概念）：

定义 2.10：称 K 关于 $(L(G), \{P_i\})(i \in I)$ 强联合正态，如果：

$$K = L(G) \cap \bigcup_i P_i^{-1}P_i(K)。$$

由强联合正态定义可得下面结论：

推论 2.5：如果 K 关于 $(L(G), \{P_i\})(i \in I)$ 强联合正态，则 K 关于 $(L(G), \{P_i\})(i \in I)$ 联合正态。

证明：因为 $K \subseteq L(G) \cap \bigcap_i P_i^{-1}P_i(K) \subseteq L(G) \cap \bigcup_i P_i^{-1}P_i(K) = K$，

所以 $K = L(G) \cap \bigcap_i P_i^{-1}P_i(K)$，得证。

这个推论说明强联合正态的条件要比联合正态苛刻一些。

定义 K 的关于 $(L(G), \{P_i\})(i \in I)$ 强联合正态子语言上界为 supCSUN(K)，则有如下性质：

推论 2.6[30]：$\sup CSUN(K) \subseteq \sup CN_i(K)$。

定理 2.9：$\sup CSUN(K) \subseteq i\sup CUN(K) \subseteq \sup CUN(K)$。

证明：显而易见，$i\sup CUN(K) \subseteq \sup CUN(K)$。

根据推论 2.6：$\sup CSUN(K) \subseteq \sup CN_i(K)$，所以：

$$\sup CSUN(K) \subseteq \bigcap_i \sup CN_i(K) = isupCUN(K)(i \in I)。得证。$$

由定理 2.9 可知，一个语言的能控且强联合正态子语言上界要小于上面求出的能控及联合正态子语言 isupCUN(K)。在无法求出 supCUN(K)的情况下，我们只能用 isupCUN(K)来生成给定语言 K 的能控和联合能观子语言。

虽然强联合正态也时常代替联合能观用来分析分散监控问题，不过从上面的论述看出，采用联合正态的概念来分析分散监控问题更好一些。而且在同样情况下，用子语言 isupCUN（K）来生成的分散监控的目标语言要比采用 supCSUN(K)要大，即对被控系统的行为的限制要小。

2.4.2 分散监控问题求解

在给出 isupCUN(K)的定义及证明前，先要给出一些语言集的定义，如下所示：

定义 2.11：$K \subseteq L(G), K = \overline{K}$，给定可观映射集合 $\{P_i\}, P_i : \Sigma^* \to \Sigma_{io}^*$。定义下面几个子语言：

$C(K) = \{M \subseteq K: M = \overline{M}$ 且 M 关于 $(L(G), \Sigma_{uc})$ 能控$\}$；

$N_i(K) = \{M \subseteq K: M = \overline{M}$ 且 M 关于 $(L(G), P_i)(i \in I)$ 正态$\}$；

$CN_i(K) = \{M \subseteq K: M = \overline{M}$，M 关于 $(L(G), P_i)(i \in I)$ 正态且 M 关于 $(L(G), \Sigma_{uc})$ 能控$\}$；

$CUN(K) = \{M \subseteq K: M = \overline{M}$，M 关于 $(L(G), \{P_i\})(i \in I)$ 联合正态且 M 关于 $(L(G), \Sigma_{uc})$ 能控$\}$。

分别用 supC(K)、sup N_i(K)、sup CN_i(K)、supCUN(K)表示语言集 C(K)、N_i(K)、CN_i(K)、CUN(K)的上界。

在后面的论述及证明中要用到以下的引理和定理：

引理 2.1：若 $L \subseteq \Sigma^*$，$L = \overline{L}$，则对 $\forall B \subseteq \Sigma^*$，$L\text{-}B\Sigma^*$ 也是闭的。

引理 2.2：如果 M、N 是闭的，则 $M \cap N$ 也是闭的。

证明：对任意串 $s \in M \cap N$，取 s 的任意子串 p，由于 $s \in M$，且 $M = \overline{M}$，所以 $p \in M$；同理 $p \in N$。则 $p \in M \cap N$；即 $M \cap N$ 中所有串的子串也在 $M \cap N$ 中。根据闭的定义得，$M \cap N$ 也是闭的。

得证。

引理 2.3：如果 M 是闭的，对可观映射 $P: \Sigma^* \to \Sigma_o^*$，$P^{-1}PM$ 也是闭的。

证明：对 $P^{-1}PM$ 中任意串 s，$\exists s' \in M, P(s) = P(s')$。取 s 的任意子串 q，则在 s' 中必能找到一个子串 q'，使得 $P(q) = P(q')$，由于 M 闭，$q' \in M$，则 $q \in P^{-1}PM$；就是说 $P^{-1}PM$ 中任意串的子串仍然属于 $P^{-1}PM$。所以 $P^{-1}PM$ 也是闭的。得证。

定理 2.10[7]：$\sup C(K) = K\text{-}D_{uc}(L(G) - B)\Sigma^*$；

$$(D_{uc}(M) = \{D_{uc}(s) : s \in M\}; D_{uc}(s) = \sup\{f: f \in \bar{s},\ f = m\sigma, \sigma \in \Sigma_c\})$$

定理 2.11[7]：$\sup N_i(K) = K\text{-}P_i^{-1}P_i(L(G) - K)\Sigma^*$；

$$\sup CN_i(K) = L(G) \cap P_i^{-1} \sup C_{pi}(P_i \sup N_i(K))$$；

（对 $D \subseteq P(L(G))$，$C_{pi}(D) = \{L \subseteq D: L$ 关于 $(P(L(G)), \Sigma_{uc})$ 能控且 L 闭$\}$）

下面给出 isupCUN(K)的定义及证明：

定理 2.12：定义 $i\sup CUN(K) = \bigcap_i \sup CN_i(K) = L(G) \cap \bigcap_i P_i^{-1} \sup C_{pi}(P_i \sup N_i(K))$，有：$isupCUN(K) \in CUN(K)$。

证明：令 $H = isupCUN(K)$。因为 $\sup C_{pi}(P_i \sup UN(K))$ 是闭的，则 $P_i^{-1}(\sup C_{pi}(P_i \sup N_i(K)))$ 也是闭的，所以根据引理 2.2，H 也是闭的。由于 $H \subseteq L(G) \cap \bigcap_i P_i^{-1}P_i \sup N_i(K) = \bigcap_i \sup N_i(K) \subseteq K$，则只要分别证明 H 是联合正态且能控：

（1） 证明 H 是联合正态的：

$$L(G) \cap \bigcap_i P_i^{-1}P_i(H) \subseteq L(G) \cap \bigcap_i P_i^{-1}P_i(P_i^{-1} \sup C_{pi}(P_i \sup N_i(K)))$$

$= L(G) \cap \bigcap_i P_i^{-1}(\sup C_{pi}(P_i \sup N_i(K))) = H$。

而显而易见的有 $L(G) \cap \bigcap_i P_i^{-1} P_i(H) \supseteq H$。

由上面两式得出 $H=L(G) \cap \bigcap_i P_i^{-1} P_i(H)$，即 H 是联合正态的。

（2） 证明 H 是能控的：令 $H_i = \sup C_{pi}(P_i \sup N_i(K))$，则 $H = L(G) \cap \bigcap_i P_i^{-1}(H_i)$

只要证明 $\forall i \in I$，$P_i^{-1} H_i$ 关于 $(L(G), \Sigma_{uc})$ 能控。

对任意 $s \in P_i^{-1} H_i$，即 $P_i(s) \in H_i$，任取不可控事件 σ，使得 $s\sigma \in L(G)$。若 $\sigma \in \Sigma_{io}$，则 $P_i(s\sigma) = P_i(s)\sigma \in H_i \Sigma_{uc}$ 且 $P_i(s\sigma) \in P_i(L(G))$；由 H_i 关于 $(P_i(L(G)), \Sigma_{uc})$ 能控，得 $H_i \Sigma_{uc} \cap P_i(L(G)) \subseteq H_i$，即 $H_i(\Sigma_{uc} \cap \Sigma_{io}) \cap P_i(L(G)) \subseteq H_i$；由此可以得出 $P_i(s\sigma) \in H_i$，即 $s\sigma \in P_i^{-1} H_i$。如果 $\sigma \notin \Sigma_{io}$，则 $P_i(s\sigma) = P_i(s)$，那么有 $P_i(s\sigma) \in H_i$，即 $s\sigma \in P_i^{-1} H_i$。综合两种情况，有 $P_i^{-1}(H_i)\Sigma_{uc} \cap L(G) \subseteq P_i^{-1} H_i$，满足能控性定义，所以 $\forall i \in I$，$P_i^{-1} H_i$ 关于 $(L(G), \Sigma_{uc})$ 能控。所以 H 关于 $(L(G), \Sigma_{uc})$ 能控。

得证。

由上面定理可得，对于任意的闭语言 $K \subseteq L(G)$，可以建立分散监控器 $\psi = (\{S_i\}, \{\varphi_i\})$，使得 $L(\Psi / G) = \text{isupCUN}(K)$。

2.4.3 例子

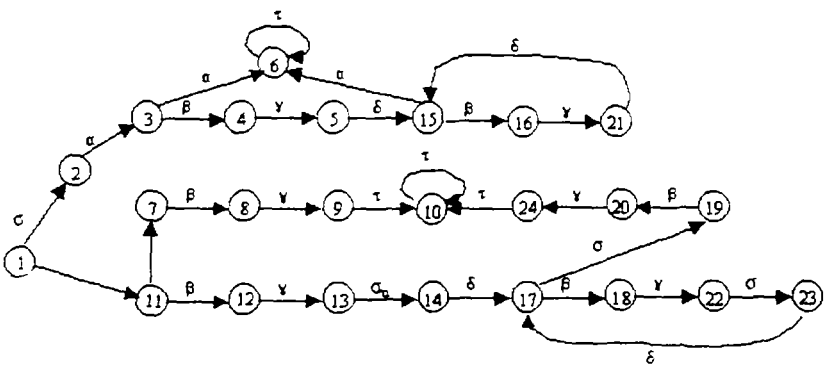


图 2.3 系统 G 的自动机模型

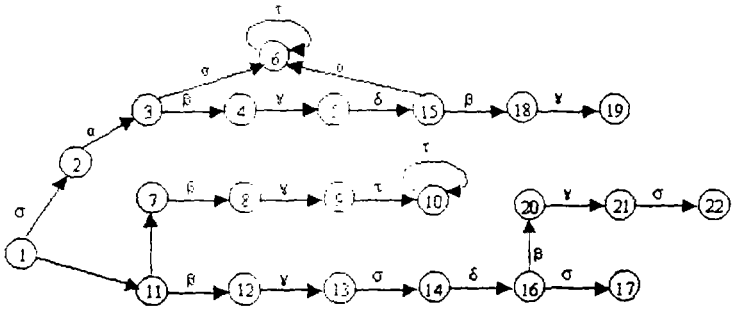


图 2.4 目标语言 K

图 2.3 给出了一个待处理的系统 G，系统中每个状态都是可标识的，定义一系列事件集合：可观事件集合 $\Sigma_o=\{\alpha, \beta, \gamma, \delta, \tau\}$；可控事件集合 $\Sigma_c=\{\alpha, \beta, \gamma, \tau\}$，对事件集合 Σ 划分 I={1,2}，对应每个本地站点。局部可观事件集 $\Sigma_{1o}=\{\alpha, \beta, \tau\}$，$\Sigma_{2o}=\{\gamma, \delta, \tau\}$。局部可控事件集合 $\Sigma_{1c}=\{\alpha, \beta, \tau\}$，$\Sigma_{2c}=\{\gamma, \tau\}$。假设有一个闭语言 K，如图 2.4 所示；要求使用分散监控将系统的行为限制在 K 以内。

我们可以按 2.4.2 中的方法来求其分散监控器，即建立各个本地站点的监控器(S_1, φ_1)和(S_2, φ_2)，使监控后系统的闭环行为 $L(\Psi/G)=\text{isupCUN}(K)$。得到的 $L(\Psi/G)$ 如图 2.5 所示，而分散监控器则如图 2.6、图 2.7 所示。

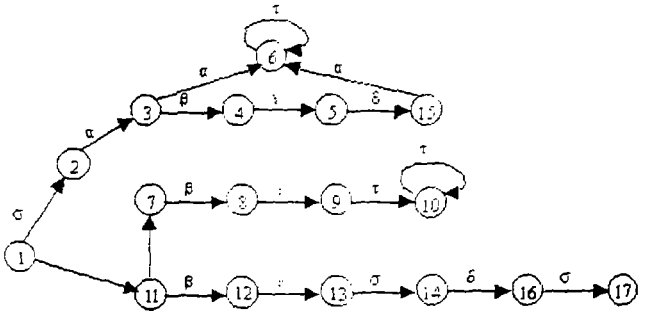
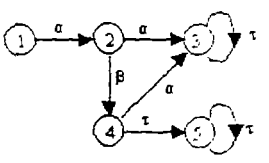


图 2.5 被控闭环行为 $L(\psi/G)$

观测器 S_1：

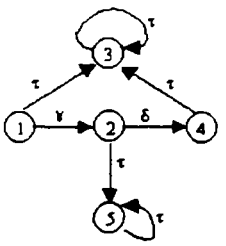


执行器 φ_1：

状态＼事件	α	β	γ	δ	τ	σ_f	σ_u
1	1	0	1	1	0	1	1
2	1	1	1	1	0	1	1
3	0	0	1	1	1	1	1
4	0	0	1	1	0	1	1
5	0	0	1	1	1	1	1

图 2.6 站点 1 的监控器 ψ_1

观测器 S_2:



执行器 φ_2:

状态 \ 事件	α	β	γ	δ	τ	σ_f	σ_u
						1	o
1	1	1	0	1	0	1	1
2	1	1	0	1	1	1	1
3	1	1	0	1	1	1	1
4	1	1	0	1	0	1	1
5	1	1	0	1	1	1	1

图 2.7 站点 2 的监控器 ψ_2

2.5 小结

本章介绍了离散事件动态系统的监控理论的一些基本知识，包括建模方法、能控性和能观性理论、监控理论等。此外，本章还论述了复杂离散事件系统的监控问题。根据被控对象和监控器配置的不同，监控问题应该采用不同的处理方法，但不论采取那种监控策略，均希望局部监控能与全局监控等效。而本章中着重论述了两种监控策略：基于局部控制目标的分散监控和基于全局目标的分散监控。

在本章的最后一节，我们讨论了基于全局目标的分散监控方法的实现问题。在该节中，通过引入了联合正态的概念来代替监控器存在条件中的联合能观性质，并提出了在给定语言 K 为闭的情况下，一个能控且联合正态子集 isupCUN(K) 的求解方法。当对给定目标语言进行分散监控时，可以用这个子集作为分散监控的目标。本文还讨论这个子集的一些性质，并通过与强联合正态的概念的比较，说明采用联合正态概念来分析问题和采用 isupCUN(K)集合来解决实际问题的优点。当然，这个子集并不一定是能控且联合正态的上界，只是次优解，这就导致过多限制系统行为，使其在实际中的应用受到影响。这些在今后研究中应该会得到解决。

第三章 DEDS 故障诊断理论

3.1 引言

离散事件系统是被描述成逻辑值的离散状态空间和事件驱动的动态系统。为了进行故障诊断，大部分大型动态系统都可以根据要被检测的故障的特性在一定程度上被抽象成离散事件系统。这样，基于 DEDS 的故障诊断方法不仅适用于离散事件动态系统（比如通信网络和计算机系统），而且适合于被视为连续变量系统并用各种的微分或者差分方程建模的传统系统。这种方法的主要优点是不需要对被诊断系统详细和深入的建模，所以适合于那些难以建模的系统。

Wonham 和 Ramage[1] 于八十年代创立了离散事件动态系统监控理论，它基于离散事件系统的自动机模型，将离散事件系统引入控制领域。该理论经过十几年的发展已日趋成熟，监控理论的研究也从纯粹的监控器设计和求解有关语言的算法中突破出来，对具有突发性、随机性和事件驱动等特性的故障系统的研究逐步展开，研究重点也由监控器设计转化为诊断器和监控器的双重设计。所以在各种 DEDS 诊断方法中，比较值得一提的就是 Sampath 等人提出的诊断器理论[48, 49]，该方法正是以监控理论作为基础的。

与其他基于 DEDS 的故障诊断方法不同的是，诊断理论的离线诊断只负责生成诊断器，并根据诊断器断定系统是否可诊断。而许多 DEDS 故障诊断方法在离线诊断中通常还需要用特定的测试序列来测试系统。诊断器理论在处理诊断问题时分两个步骤：构造被诊断系统的一个离散事件模型，然后构造一个诊断协议，即一个供诊断模块使用的规则集，来检测和识别故障。离散事件模型包含系统的正常行为和故障行为；其中，故障被表示成不可观事件，以便基于观测到的事件来推断出这些已经发生的故障。

随着研究的开展和应用领域的迫切需要，诊断器理论也得到了一定的发展，比如诊断与监控的结合[53, 64]，大规模复杂系统的诊断[61, 62, 64, 70]等方面都取得了一定的成果。本章主要介绍了诊断器理论以及以此方法为基础的活跃诊断方法。在第五章中，我们将讲述针对信息分散系统的诊断方法，并根据这些内容提出一种针对信息分散系统的活跃诊断方法；而在第六章中，则将诊断器理论运用于另一类分布式系统中，提出了一种模块化诊断方法。

3.2 诊断器理论[49]

3.2.1 系统模型

被诊断的离散事件系统是以自动机为模型的，用四元组表示：$G=(X,\Sigma,\delta,x_0)$，其中X为系统状态集；Σ为符号表，代表系统的事件集合，x_0为系统初始状态，δ为部分转移函数。系统行为由G产生的闭语言L(G)。

系统中的事件分为可观事件和不可观事件，即$\Sigma=\Sigma_o\cup\Sigma_{uo}$。对应可观映射为P。故障是用事件来表示的。设故障事件集为Σ_f，并假定故障事件都是不可观的。根据不同的故障类型将所有故障事件分成几个互不相交的非空集合：$\Sigma_f=\Sigma_{f1}\cup\Sigma_{f1}\cup\cdots\cup\Sigma_{fm}$；用$\Pi_f$表示这种划分。如果一个$F_i$类故障已经发生，就意味着集合$\Sigma_{fi}$中的某个事件发生了。诊断的目的就是要找出哪个故障集合中的事件已经发生。

对系统G，做出如下假设：

（1） 由G产生的语言L是有效的(live)。即对所有$s\in L$，存在$\sigma\in\Sigma$，使得$s\sigma\in L$。

（2） G中没有不可观事件的循环，即$\exists n_0\in N,s.t.(\forall ust\in L)\ s\in\Sigma_{uo}^{*}\Rightarrow\|s\|\le n_0$。

（$\|s\|$表示串s的长度）

3.2.2 基本概念

设 L 为一语言，用ε表示空串，对任意$s\in\Sigma^{*}$，$\bar{s}$表示 S 的闭集。下面给出一些定义：

（1）$L/s=\{t\in\Sigma^{*}\mid st\in L\}$。L/s 是事件串的集合，在这些事件串的前面加上 s 后就是 L 中的事件串。

（2）$P_L^{-1}(y)=\{s\in L:P(s)=y\}$。该事件串集合表示在 L 中与事件串 y 的可观映射相同的事件串的集合。

（3） 令s_f表示串 s 的最后一个事件，则$\Psi(\Sigma_{fi})=\{(s\sigma_f)\in L:\sigma_f\in\Sigma_{fi}\}$。即$\Psi(\Sigma_{fi})$代表的集合中的所有串的最后一个事件为属于$\Sigma_{fi}$的故障事件。

（4） $L(G, x) = \{p \in L(G): \delta(x, p)!\}$，

$L_o(G, x) = \{s \in L(G, x): s = u\sigma, u \in \Sigma_{uo}^*, \sigma \in \Sigma_o\}$，

$L_\sigma(G, x) = \{s \in L_o(G, x): s_t = \sigma\}$

可以看出，$L(G, x)$表示 G 中从状态 x 出发的所有串的集合；$L_o(G,x)$则表示 $L(G, x)$中最后一个事件为可观事件的串的集合；$L_\sigma(G,x)$则表示$L_o(G,x)$中最后一个事件为σ的串的集合。

为了构造诊断器，我们可以定义一个非确定性自动机模型 $G' = (X_o, \Sigma_o, \delta_{G'}, x_0)$。其中$\Sigma_o$和$x_0$在前面有定义；$X_o = \{x_0\} \cup \{x \in X$:存在使状态转移到 x 的可观事件$\}$。而 G′的状态转移函数为 $\delta_{G'}$，定义如下：$\delta_{G'} \subseteq (X_o \times \Sigma_o \times X_0), (x, \sigma, x') \in \delta_{G'}$，如果$\delta(x, s) = x' \wedge s \in L_\sigma(G, x)$。

由定义可得，$L(G') = P(L) = \{t: t = P(s) \wedge s \in L\}$

3.2.3 可诊断性

简单的说，对一个语言，给定一个可观事件集和一个故障的划分，当任何一类故障发生后，根据对已发生的可观事件的观测，可以在有限的延迟内检测到任何故障的发生，那么这个语言就是可诊断的。所谓有限延迟就是指从故障发生到检测到该类故障这段时间内发生的事件个数是有限的。

可诊断性的形式化定义如下：

定义 3.1[49]：假设语言 L 是闭语言且是有效的，给定可观映射 P、故障集合Σ_f和故障划分Π_f，如果满足下面条件，则 L 是可诊断的（Diagnosability）：

$$(\forall i \in \Pi_f)(\exists n_i \in N)(\forall s \in \Psi(\Sigma_{fi}))(\forall t \in L/s)(\|t\| \geq n_i \Rightarrow D)$$

其中 $D = (\forall \omega \in P_L^{-1}(P(st)))(\Sigma_{fi} \in \omega)$。

式中的$\Sigma_{fi} \in \omega$表示串ω中存在有属于Σ_{fi}的故障事件。

3.2.4 诊断器

诊断器是故障诊断的执行机构，它是基于系统模型 G 构建的。诊断器在系

统离线的时候可以用于判断系统的可诊断性。当系统在线时，则可以观测系统行为，从而执行诊断。定义故障标识集 $\Delta_f = \{F_1, F_2, \cdots, F_m\}$(故障分为 m 类)。所有可能的标识集合 $\Delta = \{N\} \cup 2^{\Delta_f}$。其中 N 表示“正常”；$F_i$ 表示该类故障已经发生。定义 $Q_0 = 2^{X_0 \times \Delta}$。

G 的诊断器 $G_d = (Q_d, \Sigma_0, \delta_d, q_0)$，其中 $q_0 = \{(x_0, \{N\})\}$；状态空间 Q_d 是 Q_0 的子集，由诊断器中从 q_0 开始在 δ_d 作用下可以达到的状态组成。Q_d 中任一状态 q_d 有如下形式：$\{(x_1, \ell_1), \cdots, (x_n, \ell_n)\}$，其中 $x_i \in X_0$，$\ell_i \in \Delta$。而 δ_d 定义如下：

$q_2 = \delta_d(q_1, \sigma) \Leftrightarrow q_2 = R(q_1, \sigma)$，其中，$\sigma \in e_d(q_1)$。下面解释一下其中几个函数的意思：

$$e_d(q_1) = \bigcup_{(x,\ell)} \{P(s) : s \in L_0(G, x)\},$$

$$R(q, \sigma) = \bigcup_{(x,\ell) \in q} \bigcup_{s \in L_0(G,x)} \{(\delta(x, s), LP(x, \ell, s))\}，\text{其中：}$$

$$LP(x, \ell, s) = \begin{cases} \{N\}，\text{如果 } \ell = \{N\} \wedge \forall i[\Sigma_{fi} \notin s]; \\ \{F_i : F_i \in \ell \vee \Sigma_{fi} \in s\}，\text{否则} \end{cases}$$

下面提供与诊断性判定有关的一些定义。

诊断器中的状态根据其故障标记可分为三类，分别定义如下：

定义 3.2：状态 $q \in G_d$，如果 $\forall (x, \ell) \in q$，$F_i \in \ell$，则称 q 是 F_i 确定（F_i-certain）的。

定义 3.3：状态 $q \in G_d$，如果 $\exists (x, \ell), (y, \ell') \in q$，使得 $F_i \in \ell$ 且 $F_i \notin \ell'$，则称 q 是 F_i 不确定（F_i–uncertain）的。

定义 3.4：对于一个状态集合 $x_1, x_2, \cdots, x_n \in X$，如果 $\exists s \in L(G, x_\ell)$，使得 $s = \sigma_1 \sigma_2 \cdots \sigma_n$，且 $\delta(x_\ell, \sigma_\ell) = x_{\ell+1 \bmod n}, \ell = 1, 2, \cdots, n$，则称该状态集合在 G_d 中组成了一个环(Cycle)。

定义 3.5：一个由 F_i 不确定的状态组成的集合 $q_1, q_2, \cdots, q_n \in Q_d$ 组成一个 F_i 不定环（F_i-indeterminate cycle），如果它满足下面条件：

（1） $q_1,q_2,\cdots,q_n$ 在 G_d 中组成一个环，即 $\delta_d(q_\ell,\sigma_\ell)=q_{\ell+1}$，$\ell=1,2,\cdots,n-1$；$\delta_d(q_n,\sigma_n)=q_1$，其中 $\sigma_\ell\in\Sigma_o$，$\ell=1,2,\cdots,n$。

（2） $\exists(x_\ell^k,p_\ell^k),(y_\ell^r,\bar{p}_\ell^r)\in q_\ell$，$\ell=1,\cdots,n$，$k=1,\cdots,m$ 且 $r=1,\cdots,m'$，使得：

(a) $\forall\ell,k,r$，有 $F_i\in p_\ell^k,F_i\notin\bar{p}_\ell^r$；

(b) 状态序列 $\{x_\ell^k\}(\ell=1,2,\cdots,n,\ k=1,\cdots,m)$ 和 $\{y_\ell^k\}(\ell=1,2,\cdots,n;r=1,\cdots,m')$ 在G'中组成一个环，即：

$$(x_\ell^k,\sigma_\ell,x_{\ell+1}^k)\in\delta_{G'},\ (x_n^k,\sigma_n,x_1^k)\in\delta_{G'},\ (x_n^m,\sigma_n,x_1^1)\in\delta_{G'},$$

$$\ell=1,2,\cdots,n-1;\ k=1,2,\cdots,m$$

$$(y_\ell^r,\sigma_\ell,y_{\ell+1}^r)\in\delta_{G'},\ (y_n^r,\sigma_n,y_1^r)\in\delta_{G'},\ (y_n^{m'},\sigma_n,y_1^1)\in\delta_{G'},$$

$$\ell=1,2,\cdots,n-1;\ r=1,2,\cdots,m'$$

根据上面的定义可以得出，诊断器的状态具有下面的属性：

引理 3.1：（1）令 $\delta_d(q_0,u)=q$，如果 q 是 F_i 确定的，则 $[\forall\omega\in P_L^{-1}(u)]\Sigma_{fi}\in\omega$。

（2）如果 $q\in Q_d$ 且 q 是 F_i 不确定的，则 $\exists s_1,s_2\in L$，使得：

$$\Sigma_{fi}\in s_1,\Sigma_{fi}\notin s_2,\quad \delta_d(q_0,P(s_1))=q \text{ 且 } \delta(x_0,s_1)\neq\delta(x_0,s_2)。$$

3.2.5 可诊断性的充要条件

如果诊断器中每一个 F_i 不确定状态必须在有限步的状态转移内进入 F_i 确定状态，则当任何故障发生后，诊断器都可以在有限步状态转移内判断该故障的发生。根据 F_i 不确定环的定义，任何 F_i 不确定环都存在 F_i 不确定状态的循环，有可能使 F_i 不确定状态永远转移不到 F_i 确定状态，所以存在 F_i 不定环的诊断器是不可诊断的。因此，根据定义 3.1、定义 3.5 及引理 3.1，可以给出可诊断性的充要条件：

定理 3.1[49]：假设语言 L 是闭语言且是有效的，L 的诊断器为 G_d，L 是可诊断的，当且仅当对任何一类故障 F_i，G_d 中不存在任何 F_i 不定环。

3.2.6 在线诊断

在构造好诊断器后，如果系统是可诊断的，那我们就可以在系统在线的时候

对其进行诊断了。系统和诊断器同时从初始状态开始运行，系统中发生的可观事件驱动着诊断器的状态发生变化。一旦诊断器进入F_i类故障的F_i确定状态，便可以断定系统已经发生了F_i类故障，这时便可以采取相应措施处理故障。

3.2.7 例子

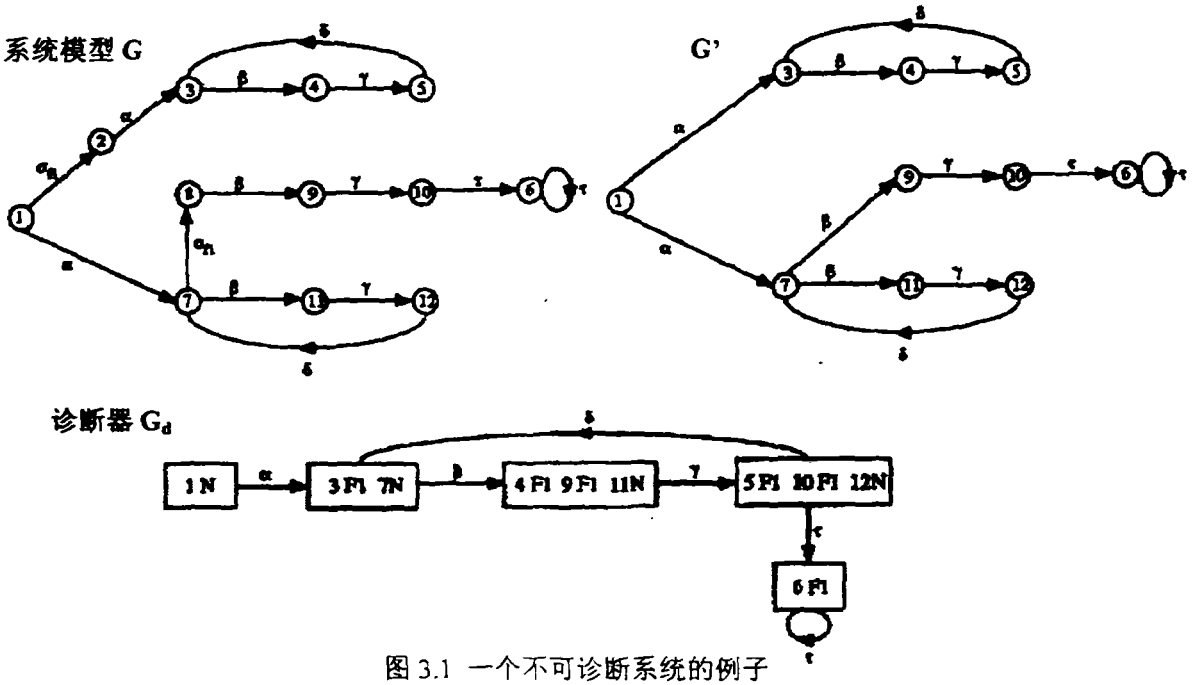


图 3.1 一个不可诊断系统的例子

图 3.1 举例说明了根据系统模型生成诊断器的过程：先按 3.2.2 从系统模型 G 得到 G'，再按 3.2.4 中的方法生成诊断器 G_d。可以看出，G_d 存在着由 {(3,{F1}),(7,{N})}、{(4,{F1}),(9,{F1}),(11,{N})}、{(5,{F1}),(10,{F1}),(12,{N})} 组成的 F1 不定环，。根据定理 3.1，系统 G 是不可诊断的。

图 3.2 则展示了另一个系统 G 及诊断器 G_d，这时从诊断器 G_d 中可以看出系统 G 是可诊断的。因此就可以使用 G_d 来诊断系统中的故障了。假设系统中产生了事件序列 $\alpha\sigma_{f1}\beta\gamma\tau\tau$，则诊断器根据观测到的事件序列 $\alpha\beta\gamma\tau\tau$ 依次经过{1N}，{3F1,7N}，{4F1,9F1,11N}，{4F1,10F1,12N}，{6F1}，{6F1}；当诊断器状态进入{6F1}时，就可以断定系统中已经发生故障，且故障的类型为 F1。而事件序列 $\alpha\beta\gamma\delta\delta$ 发生后，诊断器进入{12N}状态，这时可以断定没有发生故障。

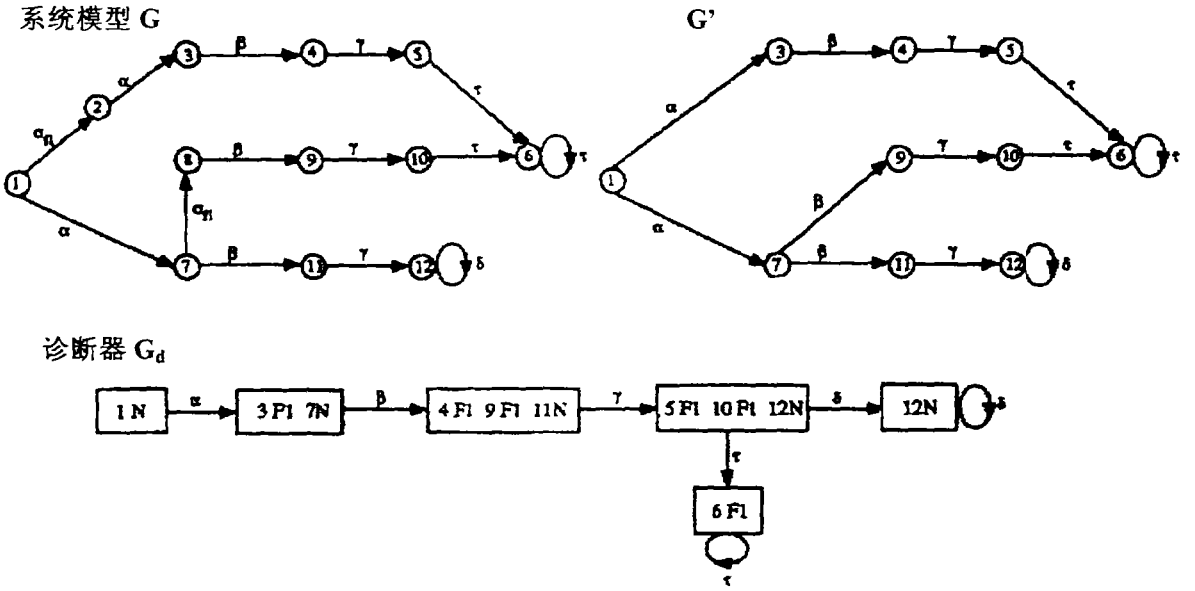


图 3.2 一个可诊断系统的例子

3.3 活跃诊断[53]

第一章简单介绍了活跃诊断的基本原理，在下面的内容中，我们将进行较为详细的叙述：

3.3.1 非有效(live)系统的可诊断性

Sampath 的诊断理论是基于语言 L 有效（live）的假设之上的，而被控系统产生的语言往往不是有效的，所以【53】中提出了一个一般性的可诊断性概念：

定义 3.6：闭语言 L 关于可观映射 P 和在故障集合 Σ_f 上的划分 Π_f 是可诊断的，如果满足下面条件：

$$(\forall i \in \Pi_f)(\exists n_i \in N)(\forall s \in \Psi(\Sigma_{fi}))(\forall t \in L/s)[((\|t\| \le n_i) \wedge (L/st = \phi) \Rightarrow D_1) \wedge (\|t\| \ge n_i) \Rightarrow D_2)$$

其中 $D_1:(\omega \in P_L^{-1}(P(st)) \wedge (L/\omega = \phi) \Rightarrow \Sigma_{fi} \in \omega;\ D_2:\omega \in P_L^{-1}(P(st)) \Rightarrow \Sigma_{fi} \in \omega$。

为了便于验证非有效语言的可诊断性，先将 L 扩展成有效语言 L^{live}。增加一个可观不可控事件 Stop，定义 $L^{live} = L_{\cup i \ge 0}\{sStop^i : L/s = \phi\}$。而所有满足 $L/s = \phi$ 的串 s 称为终结串（Terminating traces）。根据定义 3.6，有：

定理 3.2[53]：L 是可诊断的 $\Leftrightarrow$ L^{live} 是可诊断的。

根据 L^{live} 的定义和定理 3.2 可知，可以通过对 L^{live} 的诊断来实现对 L 的诊断。

3.3.2 诊断器 G_d^{live}

设 $G^{live}=(X, \Sigma\cup\{Stop\}, \delta^{live}, x_0)$，$L^{live}=L(G^{live})$。定义：

$X^{dead}=\{x\in X: \forall\sigma\in\Sigma, \delta(x, \sigma)$ 无定义$\}$。 $\delta^{live}(x, \sigma)=\delta(x, \sigma), \forall x\in X-X^{dead}$

$$\delta^{live}(x, Stop)=\begin{cases} x, & x\in X^{dead} \\ \text{无定义}, & \text{否则} \end{cases}$$

构造 G^{live} 的诊断器 $G_d^{live}=(Q_d^{live}, \Sigma_o\cup\{Stop\}, \delta_d^{live}, q_0)$，$G_d^{live}$ 称为 L 的活跃诊断器。下面来讨论一下 G_d^{live} 的构造过程及其用处：

（1）从 G_d 构造 G_d^{live}：$G_d=(Q_d, \Sigma_o, \delta_d, q_0)$，$G_d^{live}$ 与 G_d 之间不同之处在于 Stop 事件。对 $q\in Q_d$，定义

$D(q)=\{(x, \ell)\in q: (\forall u\in L(G, x))[u\in\Sigma_{uo}^{*}\wedge\delta(x, u)\in X^{dead}]\}$，

$Q^{dead}(G_d)=\{q: q\in Q_d\wedge D(q)\neq\phi\}$。

因此，对 $\forall q\in Q^{dead}(G_d)$，$\delta_d^{live}(q, Stop)=L(q)$，$\delta_d^{live}(L(q), Stop)=L(q)$

其中 $L(q)=\{(y, \ell'): (\delta^{live}(x, uStop)=y)(\ell'=\tilde{LP}(\ell, u)), (x, \ell)\in D(q), u\in\Sigma_{uo}^{*}\}$

$$\tilde{LP}(\ell, u)=\begin{cases} \ell, & \forall i(\Sigma_{f_i}\in u) \\ \ell\cup\{F_i: \Sigma_{f_i}\in u\}, & \text{否则} \end{cases}$$

此外，如果 q 满足 $\forall(x, \ell)\in q, x\in X^{dead}$，则 q=L(q)，即 $\delta_d^{live}(q, Stop)=q$。所以 G_d^{live} 与 G_d 基本相同，只是对 $\forall q\in Q^{dead}(G_d)$，或者有 $\delta_d^{live}(q, Stop)=q$，或者有 $\delta_d^{live}(q, Stop)=L(q)$，$\delta_d^{live}(L(q), Stop)=L(q)$。所以可以称 G_d^{live} 是 G_d 的有效诊断扩展。当然，G_d^{live} 也可以使用 3.2.4 中的诊断器的构造方法从 $L^{live}=L(G^{live})$ 得到。

（2）G_d^{live} 的属性

P1）设 t 为 L 中任意终结串，$p\in\tilde{t}$ 且 $p_f\in\Sigma_o$，且 $t/p\in\Sigma_{uo}^{*}$。则存在 $q'\in Q^{dead}(G_d)$，使得 $\delta_d(q_0, P(p))=\delta_d(q_0, P(t))=\delta_d^{live}(q_0, P(p))=q'$；

$\delta_d^{live}(q', Stop)=q$ 且 $\delta_d^{live}(q, Stop)=q$

P2) 设 t 为 L 中任意终结串，$\Sigma_{fi} \in t$，而 t 违反了定义 3.6 诊断性条件 D_1，则状态 $q = \delta_d^{live}(q_0, P(t\ Stop))$ 与事件 Stop 组成一个 F_i 不定环。即 $q = \delta_d^{live}(q, Stop)$ 且 q 是 F_i 不确定的。

P3）设 t 为 L 中任意终结串，$\Sigma_{fi} \in t$，而 t 不违反了定义 3.6 诊断性条件 D_1，则状态 $q = \delta_d^{live}(q_0, P(t\ Stop))$ 是 F_i 确定的。

下面的定义在活跃诊断问题解决中将会用到：

定义 3.7：令 $q' \in Q^{dead}(G_d)$，使得 $\delta_d^{live}(q', Stop) = q$; $\delta_d^{live}(q, Stop) = q$。如果状态 q 在有效诊断器 G_d^{live} 中与事件 Stop 组成一个 F_i 不定环，则状态 q' 被称为关于 (G_d, G_d^{live}) 对的 Stop 不定（Stop-indeterminate）状态。

由这些性质，并结合 3.2 中的定理可得下面推论：

推论 3.1[53]：假设语言 L 是闭语言，L 的活跃诊断器为 G_d^{live}，则

L 是可诊断的 $\Leftrightarrow$ G_d^{live} 不含任何 F_i 不确定环。

根据上面的讨论，对于非有效语言 L，L=L(G)，可以用 L 的有效诊断器 G_d^{live} 来来判断系统 G 是否是可诊断，并用它来执行系统 G 的在线诊断。

3.3.3 活跃诊断问题

定义 3.8[53]：**(活跃诊断问题，即 ADP)**假设由系统 G 产生的语言为 L，L 是正规和有效的，给定一个正规和关于（L，P）正态的语言 K，K 包含于 L，K 的所有有效子语言都是可诊断的，寻找一个 G 的监控器 S_P，使得 $L(S_P/G)=L^{act}$，并满足以下条件：

（1）$L^{act} \subseteq K$；（2）L^{act} 可诊断；（3）L^{act} 尽可能大。

ADP 问题就是要寻找 K 的一个能控、能观，可诊断的子语言上界。S_P 称为 G 的诊断控制器。从 L 得到 K 的过程见文献【53】。

下面要给出解决 ADP 过程的方法描述。从初始情况 K 开始，共分三个模块完成。模块 A 计算 K 的能控且正态子语言上界。模块 B 计算 A 中产生语言的可诊断和正态子语言上界，其方法利用了 K 的所有有效子集都是可诊断的假设及

G_d^{live} 的属性。模块 C 测试收敛性，因为经过模块 B 的计算，生成的语言未必能控，所以还要继续进行迭代计算，直至达到目的。算法大致如下：

初始化：

0-1：得到 K 的产生器 G^{legal}，即 $L(G^{legal}) = K$。

0-2：构造 G^{legal} 的诊断器 G_d^{legal}。

0-3：i=0；$H_d(0) = H_d^{live}(0) = G_d^{legal}$；$M_d(0) = L(H_d(0))$。

迭代：

模块 A：

A-1：计算 $M_d(i)$ 的关于 $(P(L)(=L(G_d)), \Sigma_{uc} \cap \Sigma_o)$ 的能控子语言上界 $M_d^{\uparrow C}(i)$。令 $M_d^{\uparrow C}(i)$ 的产生器为 $H_d^{\uparrow C}(i)$。

A-2：计算 $M(i) = P_L^{-1}(M_d^{\uparrow}(i)) = L \cap P^{-1}(M_d^{\uparrow}(i))$。令 L(H(i))=M(i)。

模块 B：

B-1：令 k=0；$\tilde{H}(0) = H(i)$；$\tilde{M}(0) = L(\tilde{H}(0)) = M(i)$；$\tilde{H}_d(0) = H_d^{\uparrow C}(i)$。

B-2：用 3.3.2 中的方法将 $\tilde{M}(k)$ 扩展成有效语言 $\tilde{M}^{live}(k)$。令 $L(\tilde{H}^{live}(k)) = \tilde{M}^{live}(k)$.用 $\tilde{H}_d^{live}(k)$ 表示 $\tilde{H}^{live}(k)$ 的诊断器。

B-3：从 $\tilde{H}_d(k)$ 中除去所有关于 $(\tilde{H}_d(k), \tilde{H}_d^{live}(k))$ 对的 Stop 不定（Stop-indeterminate）状态，其结果用 $\tilde{H}_d(k+1)$ 表示。

B-4：如果 $\tilde{H}_d(k) = \tilde{H}_d(k+1)$，则令 $H_d(i+1) = \tilde{H}_d(k)$。令 $M_d(i+1) = L(H_d(i+1))$，$H_d^{live}(k+1) = \tilde{H}_d^{live}(k)$，转到 C-1。否则令 $\tilde{M}(k+1) = P_L^{-1}(L(\tilde{H}_d(k+1)))$，且令 $\tilde{H}(k+1)$ 为 $\tilde{M}(k+1)$ 的产生器（即 $\tilde{M}(k+1) = L(\tilde{H}(k+1))$），令 k=k+1，转到 B-1。

模块 C：

C-1：如果 $H_d(i+1) = H_d(i)$，则结束。ADP 的结果是 $L^{act} = M(i)$，相应的监控器 S_P 由自动机 $H_d(i)$ 产生。否则转到 A-1。

文献中还给出了实现该算法的具体步骤，这里从略。下面的定理可以证明该

方法可以在有限步内生成 K 的一个能控、能观，可诊断的子语言上界。为了方便下一章的讨论，这里还给出了该定理的大致证明过程。

定理 3.3：该方法的迭代过程可以在有限步内收敛。M(i)是 K 的能控、能观、可诊断子语言上界。

证明：从过程中 0-1 到 A-2 可以得到：

$$M(0)=P_L^{-1}(M_d^{\uparrow C}(0))=P_L^{-1}(L(H_d(0))^{\uparrow C})=P_L^{-1}(L(G_d^{legal})^{\uparrow C})=P_L^{-1}(P(K)^{\uparrow C})=K^{\uparrow CN}$$ 。

因为 K 是正态的，所以最后一个等式可以从第二章中的定理 2.11 得到。

令 T(M(0))={M(0)中不满足可诊断性条件 D_1 的终结串}。由于 K 中任何有效子语言都是可诊断的，所以 K 中违背可诊断性定义的串一定是终结串。定义 $Pre_o(t)=s\ (t=su, s_f\in\Sigma_o, u\in\Sigma_{uo}^*)$，$Pre_o(T(M(0)))=\{Pre_o(t):t\in T(M(0))\}$。根据活跃诊断器的属性可得 T(M(0))中每一个串都会到达 $H_d^{\uparrow C}(0)=\tilde{H}_d(0)$ 中的一个 Stop 不定状态。以过程中的模块 B 中的第一次迭代为例，这次迭代移去了所有关于 $(H_d^{\uparrow C}(0),\tilde{H}_d^{live}(0))$ 的 Stop 不定状态。后续的迭代中则移去了所有关于 $(\tilde{H}_d(k),\tilde{H}_d^{live}(k))$ 的 Stop 不定状态，这些状态与可诊断性的条件不符。不失一般性，假设到收敛时得到的是 $H_d(1)$。由【53】中可以得到，模块 B 的计算过程保持语言的正态性质，而且经过 B 处理后语言是可以诊断的，所以 $P_L^{-1}(L(H_d(1))$ 是 M(0)的正态和可诊断子语言上届，但它不一定能控。如果 $P_L^{-1}(L(H_d(1))$ 不能控，则从 A-1 开始继续迭代，用 $P_L^{-1}(L(H_d(1))$ 代替 K，$H_d(1)$ 代替 $H_d(0)$。

由于 $H_d(i+1)\subseteq H_d(i),\forall i>0$ [53]。而 $H_d(0)$ 是确定性状态自动机，所以该方法可在有限步内收敛。假设该方法最后得到的语言是 M(i)，则它是正规的，因为它由确定性自动机 H(i)产生。而由该过程可得，M(i)是 K 的可诊断、能控、正态子语言上界。由于 $\Sigma_c\subseteq\Sigma_o$,且正态性和闭语言性质决定了能观性，所以 M(i)是能观的、能控、可诊断的子语言上界。而生成闭环语言 M(i)的监控器 S_P 由 $H_d(i)(=H_d^{\uparrow C}(i))$。而闭环系统的在线诊断可以由 H(i)的有效诊断器 $H_d^{live}(i)$ 来执行。得证。

该方法不但提供了一个控制器来保证闭环系统可诊断，还提供了一个诊断器供在线诊断。诊断时，被诊断系统、监控器和诊断器同时从初始状态开始运行，监控器对被诊断系统进行控制，将其行为限制在 ADP 问题的目标之内，而诊断器就对被控系统的闭环行为进行诊断，诊断方法与 3.2 中提到的在线诊断方法一致。诊断的原理图可以见图 1.2。

3.3.4 例子

下面举例说明活跃诊断方法。给定系统 G，如图 3.3 所示。L=L(G)，令 $\Sigma_f = \Sigma_{f1}=\{\sigma_{f1}\}$，$\Sigma_{uo} = \Sigma_f \cup\{\sigma_{uo}\}$，$\Sigma_{uc} = \Sigma_f \cup\{\delta\}$。图 3.3 中还给出了 G 的诊断器 G_d。

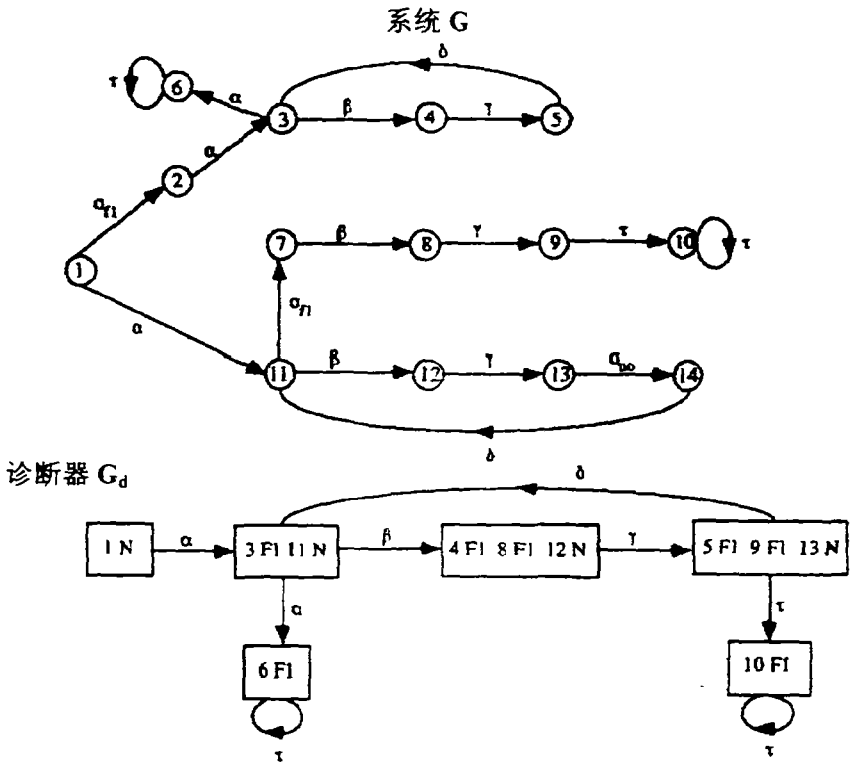


图 3.3 系统及其诊断器

可以看出，G_d 中存在一个 F1 不确定环，所以 G 是不可诊断的。

我们采用 3.3.3 中的算法来解决这个 ADP 问题。图 3.4~3.6 分别显示了这个问题的解决过程。这里说明之：

初始化：图 3.4 给出了 G^{ref} 和 G_d^{ref} 的自动机图，G^{ref} 是 G 改进后的结果，但是 $L=L(G^{ref})=L(G)$，这样改进便于生成 $K=L(G^{legal})$。K 可以通过文献中方法得到。这里 G^{legal} 与 G^{ref} 基本相同，只是去掉了从状态 21 到状态 15 的转移。而

$H_d(0) = G_d^{legal}$，如图 3.5。

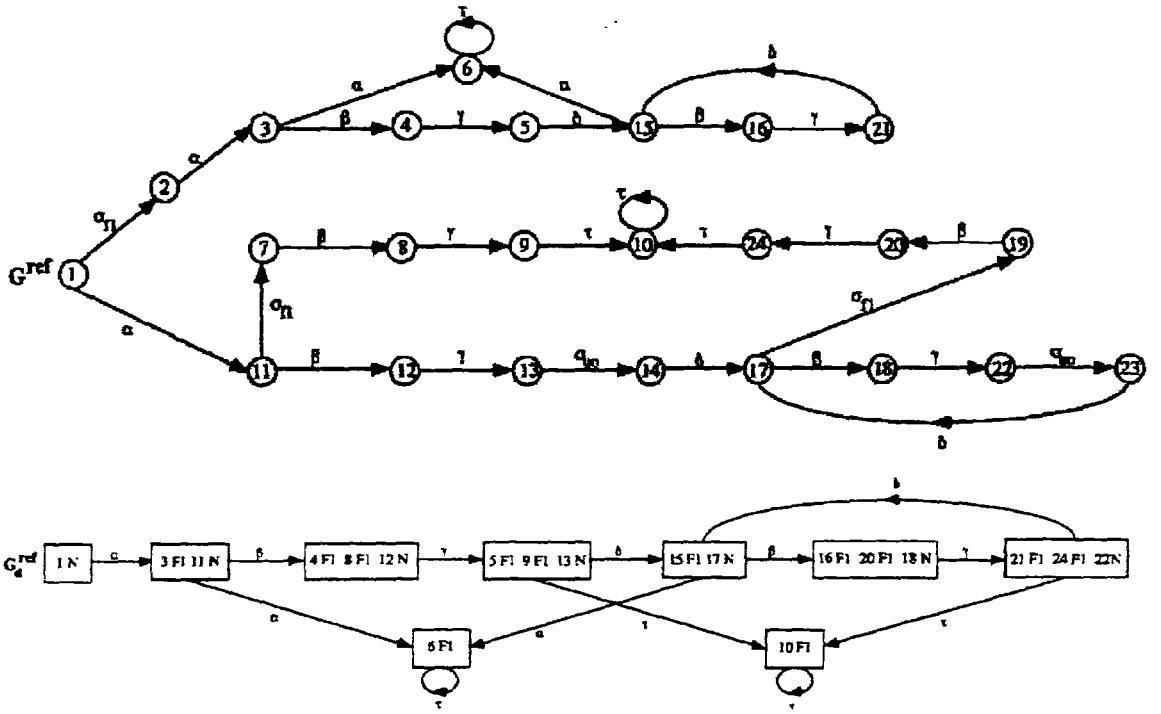
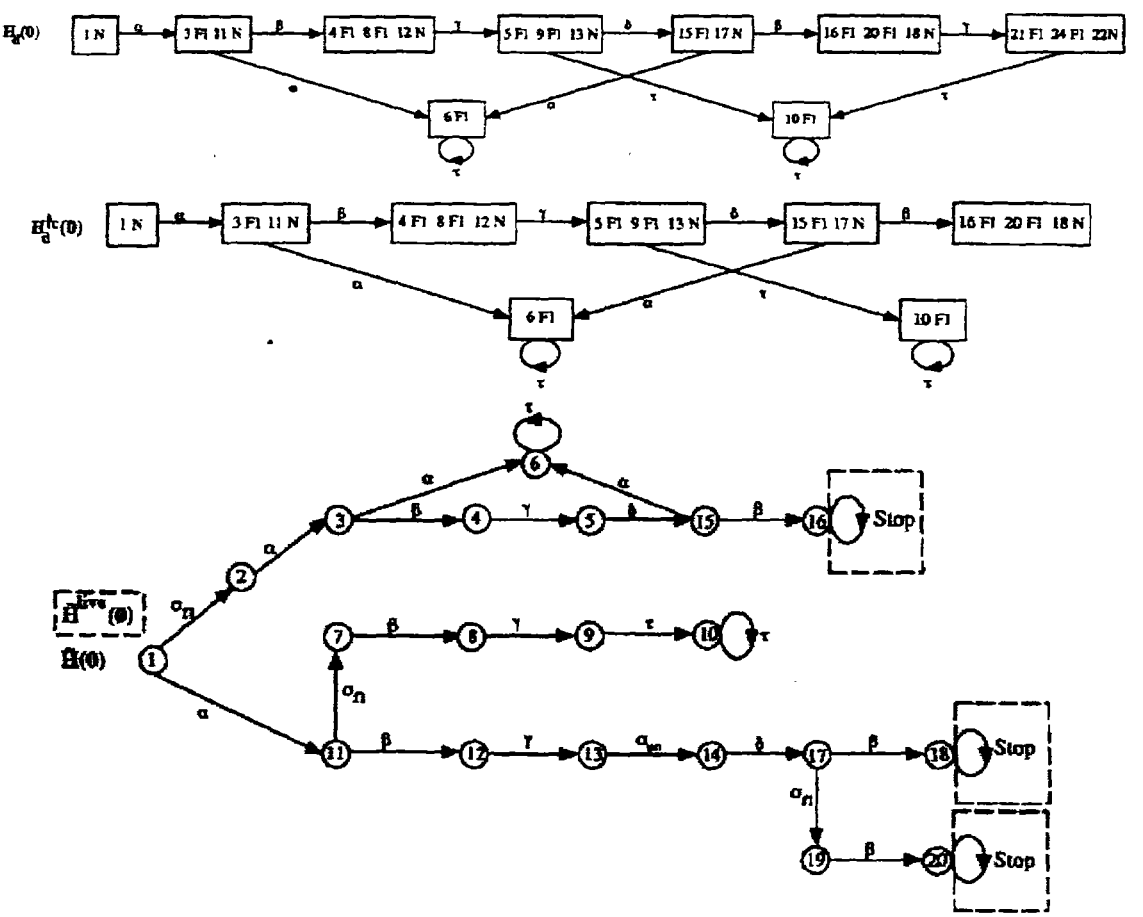


图 3.4 精练的系统模型 G^{ref} 及其诊断器 G_d^{ref}



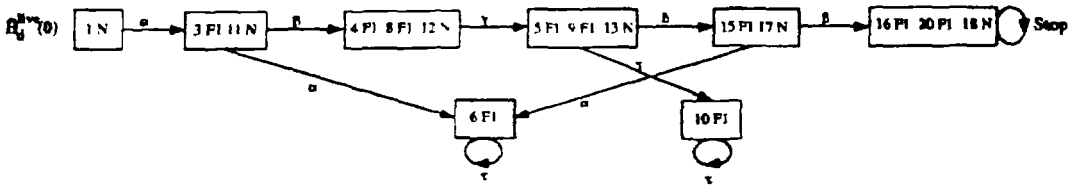


图 3.5 解决过程第一部分

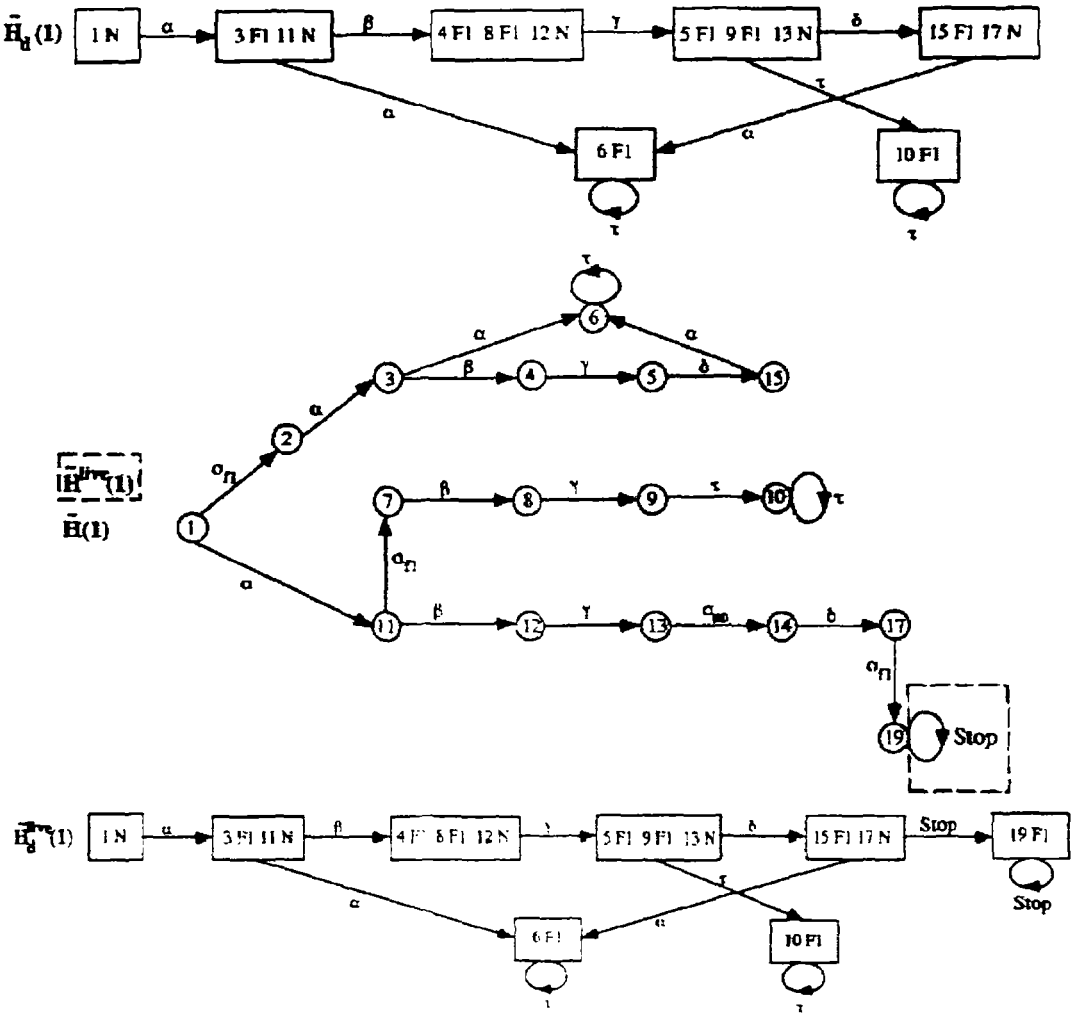


图 3.6 解决过程第二部分

第一步迭代：根据模块 A，可以从 $H_d(0)$ 得到 $H_d^{\uparrow C}(0)$，从而得到 H(0)。令 $\tilde{H}_d(0)=H_d^{\uparrow C}(0)$ 和 $\tilde{H}(0)=H(0)$，然后进入模块 B。从 $\tilde{H}(0)$ 得到其有限扩展 $\tilde{H}^{live}(0)$ 及有效诊断器 $\tilde{H}_d^{live}(0)$。这些可以见图 3.4。在排除了 $(\tilde{H}_d(0),\tilde{H}_d^{live}(0))$ 中的 stop 不定状态后，可以得到 $\tilde{H}_d(1)$。由于 $\tilde{H}_d(0)\neq\tilde{H}_d(1)$，所以要在 B 模块中再迭代一次。这时从 $\tilde{H}_d(1)$ 得到 $\tilde{H}(1)$，同样的方法从 $\tilde{H}(0)$ 得到其有限扩展 $\tilde{H}^{live}(1)$ 及有效诊断器

$\tilde{H}_d^{live}(1)$。这些可见图 3.6。由于$(\tilde{H}_d(1), \tilde{H}_d^{live}(1))$已经不含 stop 不定状态，所以得到$\tilde{H}_d(1) = \tilde{H}_d(2)$。令$H_d(1) = \tilde{H}_d(1)$，跳到模块 C 运行。由于$H_d(0) \neq H_d(1)$，所以从模块 A 开始运行，这样进入第二步迭代。

第二步迭代：$H_d(1)$能控，所以$H_d(1) = H_d^{\uparrow C}(1)$，则根据模块 A，得到 H(1)，并令$\tilde{H}_d(0) = H_d(1)$和$\tilde{H}(0) = H(1)$，然后进入模块 B。而在模块 B 的计算过程中，发现$\tilde{H}_d(0)$（第二步迭代中的）$= H_d(1) = \tilde{H}_d(1)$（第一步迭代中的），这样，$\tilde{H}_d^{live}(0)$（第二步迭代中的）$= \tilde{H}_d^{live}(1)$（第一步迭代中的）。即$(\tilde{H}_d(0), \tilde{H}_d^{live}(0))$中不含 stop 状态。所以$\tilde{H}_d(0) = \tilde{H}_d(1)$。这样，令$H_d(2) = \tilde{H}_d(0)$，然后跳到模块 C 执行。由于$H_d(2) = \tilde{H}_d(0) = H_d(1)$，则算法结束。

最后得到 ADP 的解，即$L^{act} = M(1) = L(H(1))$，而相应诊断器为$H_d(1)$。不难设计出监控器$S_p = (S, \Phi)$，其中观测器$S = H_d(1)$。通过Φ的作用将系统行为控制L^{act}内，然后再运用有效诊断器$H_d^{live}(1)$对闭环系统进行在线诊断。根据上面的讨论可知，图 3.6 中的$\tilde{H}_d(1)$、$\tilde{H}(1)$、$\tilde{H}_d^{live}(1)$分别与 S、L^{act}、$H_d^{live}(1)$的自动机模型相同。

3.4 小结

在第一章的研究领域介绍里面，我们讨论了 DEDS 故障诊断的许多理论及其方法，在这一章中，我们着重对其中的诊断器理论进行了较为详细的介绍。诊断器理论是一种基于事件的方法，即系统诊断性能不依赖于系统建模的具体结构，诊断都是通过对事件序列的分析得到的。因此这类方法具有很好的通用性。此外，本章中还对活跃诊断方法也进行了详细的讨论。活跃诊断是针对不可诊断系统的诊断问题提出的，它以 Sampath 的诊断器理论为基础，使得诊断器理论能更好的应用于实际。

上面这些都是诊断器理论的基础，笔者在它们的基础上提出了一些新的方法，这些将在后面章节进行介绍。

第四章 信息分散系统的故障诊断

4.1 引言

在第一章中，我们简要介绍了 Debouk 等人提出的针对信息分散系统的故障诊断方法，即协调分布式方法。该方法初步解决了信息分散系统的诊断问题。不过这种方法要首先判断系统是否可诊断，不能应用于不可诊断系统。在现实中，很少有系统本身是可诊断的，所以为不可诊断的系统设计诊断器显得尤为重要。第三章提到的活跃诊断方法能很好的解决这一问题，不过只是针对集中式诊断的情况。因此我们根据需要对其加以改造，以适应信息分散系统的诊断需要。由于 Debouk 的方法中所讨论的系统是信息分散系统，所以最好使用第二章所讨论的分散监控的方法来控制系统的行为，而监控器同时可以用作在线诊断。

基于上面的考虑，本章提出了一种对信息分散系统的活跃诊断的方法，称之为分散式活跃诊断方法。其体系结构与 Debouk 的基本相同，只是在本地的观测器上附加了控制器来控制整个系统的行为，同时用本地诊断器对被控行为进行诊断，并将诊断信息送到协调器，以产生全局的诊断信息。为了简便起见，只讨论了使用了协议 1 来进行诊断的方法，而其他协议稍加改变也可以应用。诊断分离线诊断和在线诊断部分：离线诊断生成系统的分散监控器和诊断器，在线诊断利用协议对被控闭环系统行为进行诊断。

本章首先比较详细的介绍协调分布式方法，然后再分三个部分来介绍分散式活跃诊断方法：首先给出了这类问题的描述；其次讨论了离线诊断部分，包括分散监控器和诊断器的生成方法；接下来就给出了在线诊断方法，主要是使用 Debouk 方法中的协议 1。在本章的最后，我们将给出一些结论。

4.2 协调分布式故障诊断方法[61, 62]

4.2.1 协调分散结构

协调分布式故障诊断方法是由 Debouk 等人所提出的，它以 Sampath 的诊断器理论为基础，其体系结构如图 1.3 所示。这种结构适合处理信息分散系统，即全局系统的信息分布在各个站点。该结构包含一系列本地站点、协调器以及故障

发现模块。这里着重论述两个站点的情况。

在这种体系结构中，顶层模块代表系统模型 G，被诊断系统模型介绍可以见 3.2。每个站点有两个模块：观察模块和诊断模块。站点 i 基于它可运用的检测能力对系统进行观察，假设它能观察到的事件集合为 $\Sigma_{io} \subseteq \Sigma_o$，而 $\bigcup_i \Sigma_{io} = \Sigma_o (i=1,2)$。令每个站点的可观映射为 $P_i : \Sigma^* \to \Sigma_{io}^*$。每个站点根据自己所观察到的事件序列进行诊断，产生各自的诊断信息；然后将各自的诊断信息以一定的形式送到协调器。通信中传递的信息类型是由通信规则决定的。协调器根据某些决策规则，从送来的信息中判定故障发生情况。如果检测到故障发生，就交给故障发现模块处理。

在下面假设的情况下，将分析系统的诊断性能：

A1. L(G)是有效的。

A2. G 中没有任何关于 Σ_{io}（i=1,2）的不可观事件串。

A3. 对每个站点的诊断来说，L(G)是不可诊断的。

A4. 每个站点和协调器之间的通信都是可靠的，即所有从一个站点发出的信息都会正确有序的被协调器接收。

A5. 所有站点和协调器之间的通信的信息接收时的顺序与发送时相同，即通信是全局有序的。

A6. 每个站点的可观事件集合对所有站点来说都是普遍的知识。

A7. 各个站点向协调器提交的可以只是一些加工过的原始数据的。

A8. 协调器没有整个系统的模型，即它不知道整个系统的动态行为。协调器的存贮容量和处理能力都是有限的，所以结构很简单。

假设 A1 保证系统不会死循环，根据前面两节内容，可以适当放宽。假设 2 与 Sampath 理论中假设 A2 相同。A3 排除了极端的情况，在这种情况下，分散诊断就没有意义了。A5 保证协调器收到的所有信息的全局顺序是正确的。A4、A6、A7 都是自我解释的。A8 与等级体系结构的特点有关。

4.2.2 可诊断性定义

第三章中的诊断性定义（定义 3.1）只适用于集中式系统，不能直接运用于协调分散系统。对于协调分散系统的可诊断性进行定义就必须考虑到用于产生本地诊断信息、通信规则与决定规则的规则。在前面的介绍中，可以看出每个站点

都只和协调器进行通信。设 C 为协调器的诊断信息。

定义 4.1：协调器的诊断信息 C 是 F_i 确定的，如果基于 C，协调器可以肯定故障 F_i 已经发生。

定义 4.2：在前面描述的协调分散结构下，协议被定义为一个包含下面几个方面的规则：本地诊断信息的产生、站点用于同协调器进行通信所用的规则、协调器所用的决策规则。

根据上面两个定义，可以定义给定协议下的可诊断性：

定义 4.3[61]：对于一个闭且有效的语言 L，给定协议 A，可观映射集合 P_1、P_2 和一个基于故障集合 Σ_f 的故障划分 Π_f，L 在 A，P_i 和 Π_f 的情况下可诊断，如果它满足下面条件：

$$(\forall i \in \Pi_f)(\exists n_i \in N)(\forall s \in \Psi(\Sigma_{fi}))(\forall t \in L/s)(\|t\| \geq n_i \Rightarrow C\text{是}F_i\text{确定的})。$$

这个定义要求协调器在任何故障发生后一个有限得延迟内必须能检测到。

4.2.3 分散式诊断的实现目的

任何协调分散结构的实现性能都不会超过集中式系统的，所以实现这种结构的目的是可以诊断集中式诊断器所能诊断的所有故障类型，即性能上与集中式诊断器相同。

4.2.4 协议 1 描述

4.2.4.1 本地点诊断信息

本地站点的诊断信息是由下面定义的扩展诊断器产生的。G 的扩展诊断器定义为 $G_d^e = (Q_d^e, \Sigma_o, \delta_d^e, q_0^e)$。这四个元素分别代表状态集合，事件集合，转移函数和初始状态。定义 $q_0^e = \{((x_0, \{N\}), (x_0, \{N\}))\}$，每个状态 $q \in Q_d^e$ 的形式是：

$$q = \{((x_1, \ell_1), (x_1', \ell_1')), ((x_2, \ell_2), (x_2', \ell_2')), \cdots, ((x_n, \ell_n), (x_n', \ell_n'))\}$$

每个 (x, ℓ) 对都在 Q_o 中，即 $x \in X_o$，$l \in \Delta$。（这些符号的定义可见第三章 3.2）而 q 中每一个对 $((x_i, \ell_i), (x_i', \ell_i'))$ 表示：x_i' 表示在一个可观事件发生后对系统状态的一个估计，而 ℓ_i' 是相应故障标志；x_i 表示在 G 中的 x_i' 的直接前任状态（即 $\exists \sigma \in \Sigma_o, \delta_G'(x_i, \sigma) = x_i'$）、而 ℓ_i 是相应故障标志。所以 $Q_d^e \subseteq Q_o \times Q_o$。$\delta_d^e$ 的构造方

法与诊断器转移函数δ_d的构造方法相似，只是出现在G_d中的G的每个状态都要带上它们在G'中的直接前任状态，同时还要加上相应的故障标志。而标识传播规则可见3.2.4的诊断器构造方法。由构造过程可知：$L(G_d^e)=L(G_d)=P(L)$。

下面给出一些与后面讨论有关的定义：

定义 4.4：定义状态映射$SP:Q_o\times Q_o\to Q_o$：

$q=\{((x_1,\ell_1),(x_1',\ell_1')),((x_2,\ell_2),(x_2',\ell_2')),\quad,\cdots,((x_n,\ell_n),(x_n',\ell_n'))\}\ \mapsto SP(q)=\{(x_1',\ell_1'),(x_2',\ell_2'),\cdots,(x_n',\ell_n')\}$。

将定义4.4加以扩展，可得$SP(G_d^e)=G_d$。

定义 4.5：令$q=\{((x_1,\ell_1),(x_1',\ell_1')),((x_2,\ell_2),(x_2',\ell_2')),\cdots,((x_n,\ell_n),(x_n',\ell_n'))\}$为扩展诊断器$G_{dj}^e$的一个状态，$j\in\{1,2\}$。定义集合：

$$S_j(q)=\{s\in(\Sigma\setminus\Sigma_{oj})^*:s\in L_\sigma(G,x_k'),\text{其中}\sigma\in\Sigma_{oi}(i\in\{1,2\}/\{j\}),k\in\{1,\cdots,n\}\}$$

而关于$\Sigma\setminus\Sigma_{oj}$的q的不可观到达集合定义如下：

$$UR_j(q)=\{q\}\cup\{((y_s,\ell_s),(y_s',\ell_s'))\}$$

其中y_s'是q中某些状态$x_k'(k\in\{1,\cdots,n\})$的后继，满足$\delta(x_k',s)=y_s',s\in S_j(q)$。$y_s$为在$G'$中的$y_s'$的直接前任状态，$\ell_s'$、$\ell_s$为相应状态的故障标志。

定义 4.6：G_d^e中状态q是F_i确定的，如果$\forall(x,\ell)\in SP(q),F_i\in\ell$。

定义 4.7：G_d^e中状态q是F_i不确定的，如果$\exists(x,\ell),(y,\ell')\in SP(q),F_i\in\ell$且$F_i\notin\ell'$。

定义 4.8：对于一个状态集合$q_1,q_2,\cdots,q_n\in Q_d^e$，如果$\delta_d^e(q_\ell,\sigma_\ell)=q_{((\ell+1)\bmod n}$，$\ell=1,2,\cdots,n$则称该状态集合在$G_d^e$中组成了一个环(Cycle)。

定义 4.9：Q_o中一个状态集合$\{(x_i,\ell_i)\}(i\in1,2,\cdots,n)$在$G_d^e$中组成一个匹配环，如果$\exists q_i\in Q_d^e(i=1,2,\cdots,n)$，使得：

$((x_i,\ell_i),(x_{i+1},\ell_{i+1}))\in q_{i+1},i=1,2,\cdots,n\text{-}1;((x_n,\ell_n),(x_1,\ell_1))\in q_{i+1}$。

定义 4.10：Q_d^e中一个状态集合$\{q_i\}(i\in1,2,\cdots,n)$在$Q_d^e$中组成一个$F_i$不定环，

如果满足下面条件：（1）$\exists\{(x_j, \ell_j)\}(j=1,2,\cdots,n)$，使得对每一个 j，都有 $(x_j, \ell_j)\in SP(q_j)$，$\{(x_j, \ell_j)\}$ 在 G_d^e 中组成一个匹配环，且 $F_i\in\ell_j$；（2）$\exists\{(y_j, \ell_j')\}(j=1,2,\cdots,n)$，使得对每一个 j，都有 $(y_j, \ell_j')\in SP(q_j)$，$\{(y_j, \ell_j')\}$ 在 G_d^e 中组成一个匹配环，且 $F_i\notin\ell_j'$。

属性 4.1[61]：给定系统 G，诊断器 G_d 和扩展诊断器 G_d^e，G_d 中存在 F_i 不定环当且仅当 G_d^e 中存在 F_i 不定环。

根据属性 4.1 有下面定理：

定理 4.1[61]：对一个闭且有效的语言 L，可观映射 P 和一个基于故障集合 Σ_f 的故障划分 Π_f，L 在 P 和 Π_f 的情况下可诊断当且仅当它的扩展诊断器 G_d^e 对于任何故障 F_i，都不含 F_i 不定环。

4.2.4.2 通信规则

定义通信规则 **CR:=(CR1,CR2)**如下：

[CRi], i=1,2：站点 i 观测到事件 $\sigma\in\Sigma_{oi}$ 后，它将相应的扩展诊断器 G_{di}^e 的相应状态 q_i、关于 $\Sigma\setminus\Sigma_{oj}$ 的 q_i 的不可观到达集合 $UR_i(q_i)$、标志位 SB_i（如果 $\sigma\in\Sigma_{oj}, j\in\{1,2\}, j\neq i$，则 $SB_i=1$；如果 $\sigma\in\Sigma_{oj}$，则 $SB_i=0$）这三样内容传到协调器。

4.2.4.3 决策规则

协调器的决策规则包括两个方面：（1）更新协调器信息的规则；（2）判定故障发生并传递给故障发现模块的规则。下面分别叙述之：

定义 **4.11**：令 $q_1, q_2\in Q_o\times Q_c$，其中：

$$q_1=\{((x_1, \ell_1), (x_1', \ell_1')), ((x_2, \ell_2), (x_2', \ell_2')), \cdots, ((x_n, \ell_n), (x_n', \ell_n'))\},$$

$$q_2=\{((y_1, \ell_1), (y_1', \ell_1')), ((y_2, \ell_2), (y_2', \ell_2')), \cdots, ((y_m, \ell_m), (y_m', \ell_m'))\}$$

定义 $\cap_e^i, i\in\{L,R\}$ 为作用于 q_1、q_2 上的一种交运算：

$q_1\cap_e^i q_2=\{((z, \ell), (z', \ell'))\in Q_o\times Q_o$：对 $i\in\{1,2,\cdots,n\}, j\in\{1,2,\cdots,m\}$，如果

$(x_i', \ell_i') = (y_j', \ell_j')$，有$(z', \ell') = (x_i', \ell_i') = (y_j', \ell_j')$；如果$i = L$, $(z, \ell) = (x_i, \ell_i)$，否则$(z, \ell) = (y_j, \ell_j)\}$。

定义 4.12：令$q_1, q_0 \in Q_o \times Q_o$，其中：

$$q_1 = \{((x_1, \ell_1), (x_1', \ell_1')), ((x_2, \ell_2), (x_2', \ell_2')), \cdots, ((x_n, \ell_n), (x_n', \ell_n'))\},$$

$$q_0 = \{((y_1, \ell_1), (y_1', \ell_1')), ((y_2, \ell_2), (y_2', \ell_2')), \cdots, ((y_m, \ell_m), (y_m', \ell_m'))\}$$

定义$\cap_c, i \in \{L, R\}$为作用于q_1、q_2上的另一种交运算：

$q_1 \cap_c q_0 = \{((z, \ell), (z', \ell')) \in Q_o \times Q_o$：对 $i \in \{1,2,\cdots,n\}, j \in \{1,2,\cdots,m\}$，如果$(x_i, \ell_i) = (y_j', \ell_j')$，有$(z, \ell) = (x_i, \ell_i) = (y_j', \ell_j')$，$(z', \ell') = (x_i', \ell_i')\}$

表 4.1 协议 1 中协调器的信息更新规则

当最后收到的信息是从G_{d1}^e发来的

	SB	SB1	C	新的 SB	新的 SB_{1old}	新的 SB_{2old}
DR1	0	0	$(R1 \cap_e' R4) \cap_c C_{old}$	0	1	0
DR2	0	1	等待	1	不变	不变
—	1	0	不可能	—	—	—
DR3	1	1	$(R1 \cap_e' R2) \cap_c C_{old}$	0	1	1

当最后收到的信息是从G_{d2}^e发来的

	SB	SB2	C	新的 SB	新的 SB_{1old}	新的 SB_{2old}
DR4	0	0	$(R2 \cap_e' R3) \cap_c C_{old}$	0	0	1
DR5	0	1	等待	1	不变	不变
—	1	0	不可能	—	—	—
DR6	1	1	$(R1 \cap_e' R2) \cap_c C_{old}$	0	1	1

协调器将当前诊断信息存在寄存器 C 中，此外还有 8 个寄存器用来存储信息以及与更新信息有关的过去的一些相关信息。这些存储器为：R1，R2，R3，R4，C_{old}，SB，SB_{1old}和 SB_{2old}。R1 和 R2 分别存放G_{d1}^e和G_{d2}^e的最新状态，R3 和 R4 分别存放G_{d1}^e和G_{d2}^e的最新不可观可达。状态C_{old}存放先前的协调器诊断信

息，SB 用于判断最后一个可观事件是被所有站点都观测到（1）还是没有（0）。SB_{1old} 和 SB_{2old} 提供计算新的协调器诊断信息所必须的信息。表 4.1 给出了信息更新规则。由表 4.1 可知，当前的 SB_{iold} 决定了 $\cap_e^i$ 中 i 的值：如果 SB_{iold} =1，那么 i=L；否则 i=R。当收到从 G_{di}^e 来的消息时，根据 SB 与 SB1 的值决定应采取 DR1~DR3 中的哪一个规则，来生成新的 C，SB，SB_{1old} 和 SB_{2old}。一旦 C 中的值是 F_i 确定的，协调器便通知故障发现模块故障 F_i 已经发生。

4.2.4.4 诊断属性

属性 4.2[61]：假设当系统执行过事件序列 $s=s_1aub(a, b\in\Sigma_o(=\Sigma_{o1}\cup\Sigma_{o2}), u\in\Sigma_{uo}^*)$ 后，扩展诊断器 G_{d1}^e，G_{d2}^e 和 G_d^e 中的状态分别是 q_1，q_2，q。而经过串 s_1a 后，G_d^e 的状态是 q_{old}。下面的一些命题是成立的：

（1）如果 $b\in\Sigma_{o1}\cap\Sigma_{o2}$，则（i）$(q_1\cap_e^L q_2)\cap_c C_{old}$，如果 $a\in\Sigma_{o1}$；

（ii）$(q_1\cap_e^R q_2)\cap_c C_{old}$，否则。

（2）如果 $b\in\Sigma_{o1}\backslash\Sigma_{o2}$，则 （i）$(q_1\cap_e^L UR_2(q_2))\cap_c C_{old}$，如果 $a\in\Sigma_{o1}$；

（ii）$(q_1\cap_e^R UR_2(q_2))\cap_c C_{old}$，否则。

（3）如果 $b\in\Sigma_{o2}\backslash\Sigma_{o1}$，则（i）$(UR_1(q_1)\cap_e^L q_2)\cap_c C_{old}$，如果 $a\in\Sigma_{o1}$；

（ii）$(UR_1(q_1)\cap_e^R q_2)\cap_c C_{old}$，否则。

属性 4.3[61]：（1）在协议 1 下协调器的诊断信息 C 内容与集中式扩展诊断器 G_d^e 相同的状态。（2）协议 1 可以达到和集中式诊断器相同的诊断性能。

定理 4.2[61]：对一个闭且有效的语言 L，给定可观映射集合 P_1、P_2 和一个基于故障集合 Σ_f 的故障划分 Π_f，L 在协议 1，P_i 和 Π_f 的情况下可诊断的充要条件式诊断器 G_d 无 F_i 不定环。

由上面讨论可知，任意信息分散系统只要符合 4.2.1 中的假设条件，并满足定义 3.1 的可诊断性条件，就可以使用协议 1 进行分散诊断。除了协议 1 外，还有两种用于诊断的协议，所需资源小于协议 1，但必须在增加限制条件的前提下才能达到集中式诊断器的性能，详细情况见文献【61】。

4.2.5 讨论

4.2.5.1 扩展到多个站点的情况

如果将协议 1 扩展到多个站点，通信协议是不需要改变的，只要将决定规则用下面的方法加以扩展：如果遇到的事件对于所有站点都是可观的，则将各个站点扩展诊断器的当前状态进行 $\cap_e'$ 运算；否则对可观的站点取当前状态，不可观的取当前状态的不可观到达，然后进行 $\cap_e'$ 运算。将得到的结果与 q_{old} 求 $\cap_e$ 运算。这样所进行的诊断的性能仍然与集中式诊断相同。

4.2.5.2 发生通信时延的情况

通信过程中经常会由于时延的原因，导致接收到的信息顺序与发送的不相同，即系统的假设条件 A4 与 A5 不满足。在这种情况下，在接收到一系列信息后，列出这些信息可能的所有排列顺序并分类，并将决策规则应用于所有可能的信息序列，然后去掉那些应用决策规则后不符合要求的结果。文献【62】对此进行了较为详细的讨论。文中还提出了两种改进的方法：（1）各个本地诊断器等到对所有诊断器都可观测的事件发生后，再把产生的诊断信息送往协调器。（2）协调器依次访问各个站点，取得该站点的诊断信息；不过有个前提条件，就是协调器必须再系统执行新的可观事情之前得到轮巡信息。通过采用这些方法，较好地解决了乱序的问题，不过因此也使得诊断产生的时间变长，协调器的存储要求增加。文献【62】中是以协议 2 为例进行说明的，当然其他协议也可运用这种方法解决乱序问题。

4.2.6 例子

给定一个系统，如图 4.1 所示，共有两个站点。其中 $\Sigma=\{a,b,c,d,e,\sigma\}$，$\Sigma_{uo}=\{\sigma\}$，$\Sigma_{f1}=\{\sigma\}$，$\Sigma_{o1}=\{a,c,d,e\}$，$\Sigma_{o2}=\{b,d,e\}$。图 4.2 给出了系统的诊断器 G_d 及扩展诊断器 G_d^e。由图 4.2 可以得知，G 是可诊断的，由定理 4.2 可知，G 在协议 1 下也是可诊断的。图 4.3 给出了两个站点的本地扩展诊断器 G_{d1}^e、G_{d2}^e。而运用协议 1 就能对系统进行分散诊断了。表 4.2 给出了当系统产生了一个事件序列 abcd 时，则采用协议 1 进行诊断的过程。

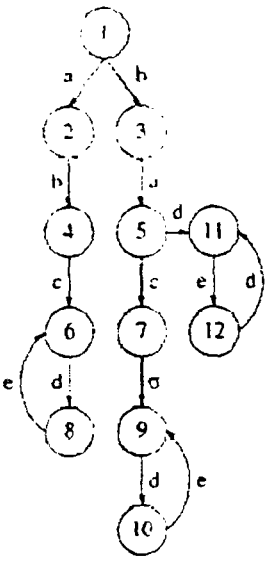


图 4.1 例子中的系统 G

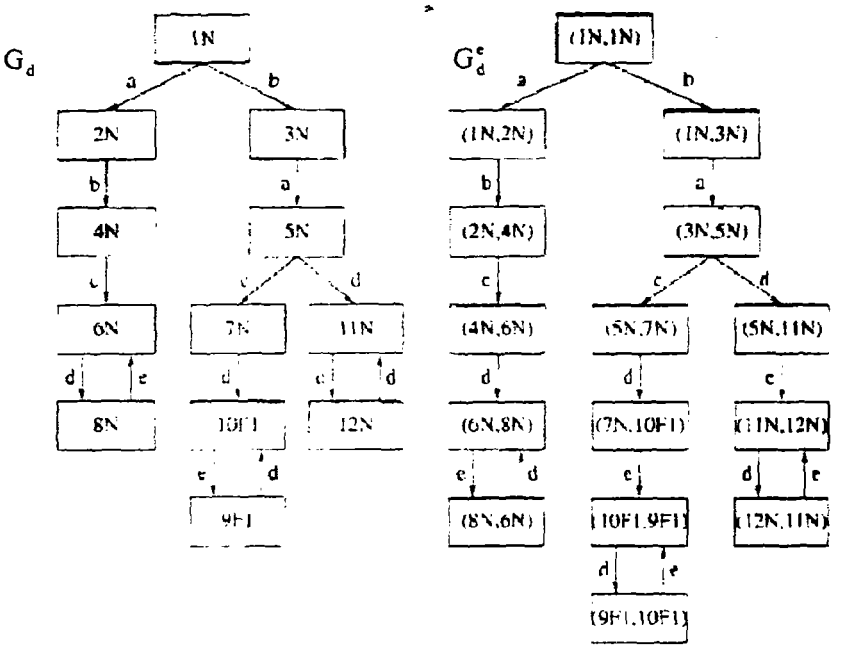


图 4.2 诊断器 G_d 和扩展诊断器 G_d^e

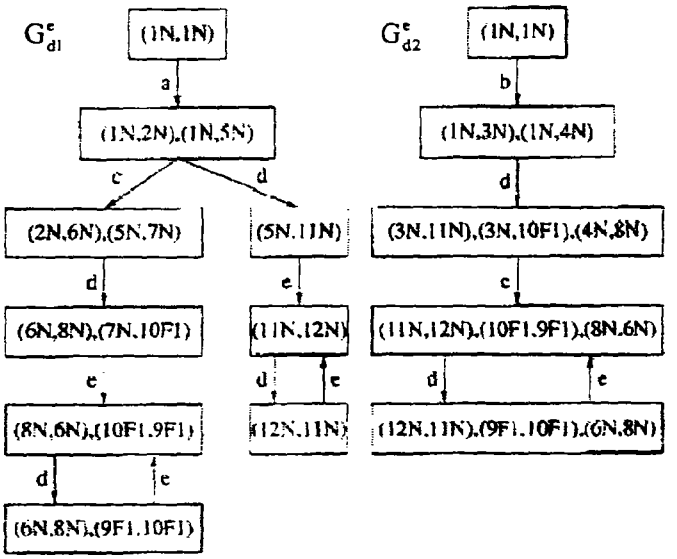


图 4.3 两个站点的扩展诊断器 G^e_{d1} G^e_{d2}

事件	R1	R3	R2	R4	C_{old}	C
ε	(1N,1N)	(1N,1N) (1N,3N)	(1N,1N)	(1N,1N) (1N,2N)	(1N,1N)	(1N,1N)
a	(1N,2N) (1N,5N)	(1N,2N) (1N,5N) (2N,4N)	(1N,1N)	(1N,1N) (1N,2N)	(1N,1N)	(1N,2N)
b	(1N,2N) (1N,5N)	(1N,2N) (1N,5N) (2N,4N)	(1N,3N) (1N,4N)	(1N,3N) (1N,4N) (3N,5N) (5N,7N) (4N,6N)	(1N,2N)	(2N,4N)
c	(2N,6N) (5N,7N)	(2N,6N) (5N,7N)	(1N,3N) (1N,4N)	(1N,3N) (1N,4N) (3N,5N) (5N,7N) (4N,6N)	(2N,4N)	(4N,6N)
d	(6N,8N) (7N,10F1)	(6N,8N) (7N,10F1)	(3N,11N) (3N,10F1) (4N,8N)	(3N,11N) (3N,10F1) (4N,8N)	(4N,6N)	(6N,8N)

表 4.2 运用协议 1 诊断过程的分析

4.3 分散式活跃诊断方法

4.3.1 问题描述

给定一个信息分散系统，系统的自动机模型为 $G=(Q, \Sigma, \delta_c, q_0, Q_m)$，可控事件集合 Σ_c，可观事件集合 Σ_o，$\Sigma_c \subseteq \Sigma_o$，一个对事件集合 Σ 划分 $I=\{1,2,...n\}$，对应每个本地站点。对每个站点，有局部事件集 Σ_i、局部可观事件集 Σ_{io}、局部可控事件集 Σ_{ic} $(i \in I)$，满足 $\Sigma=\bigcup_i \Sigma_i$，$\Sigma_c=\bigcup_i \Sigma_{ic}$，$\Sigma_o=\bigcup_i \Sigma_{io}$，$\Sigma_{ic} \subseteq \Sigma_{io} \subseteq \Sigma_i$，而且各个划分可能有共同事件。该系统的故障事件集为 Σ_f，故障划分 Π_f 将所有故障事件分成几个互不相交的非空集合：$\Sigma_f=\Sigma_{f1} \cup \Sigma_{f2} \cup ... \cup \Sigma_{fm}$。。设映射 $P_i: \Sigma \to \Sigma_{io}$，将其扩展为 $P_i: \Sigma^* \to \Sigma_{io}^*$。系统本身不一定可以诊断，要求在各个站点对系统进行分散式诊断。

在 3.3 中提到，活跃诊断的核心就是生成给定系统 G 的能控、能观、可诊断子语言上界 L^{act}。能控、能观是监控器存在的条件，可诊断性则保证被控闭环系统行为可以诊断。但在 3.3 中，L^{act} 只能用集中式监控器来实现，因为 L 不满足分散监控器存在的条件，即联合能观。所以，解决信息分散系统的活跃诊断问题的关键一步就是求出 G 的一个能控、联合能观、可诊断子语言。由文献【61】可知，只要系统是可诊断的，就可以运用分散诊断方法进行诊断。所以，针对信息分散系统的活跃诊断可分下面三个步骤：

（1） 求出给定系统 G 的一个能控、联合能观、可诊断子语言 K。

（2） 从 K 得到相应的分散监控器和本地诊断器。

（3） 利用 4.2 中的方法进行分散诊断。

其中，（1）和（2）构成了方法中的离线诊断部分，（3）则是在线诊断。

4.3.2 离线诊断

上面描述的系 G 可能不可诊断，所以要对其进行诊断，就必须将系统行为限制在可诊断范围内，这就要用到活跃诊断。而 3.3 中讨论的活跃诊断是用集中式监控实现的，所以必须对该理论进行一些修改，使之能采用分散式监控。而离线诊断就是要生成分散监控器和诊断器，关键就是找到闭的、关于 $(L(G), \Sigma_{uc})$ 能

控、关于$(L(G), \{P_i\})$联合能观的且是可诊断的语言 K，这个问题可以用改进的 ADP 问题来描述：

修改 ADP 问题如下：

Decentralized Active Diagnosis Problem：给定一个系统 G，令 L(G)=L，L 是正则且正规的语言。给定一个如 4.3.1 所描述的分散监控结构，并给定一个正则且闭的语言$K \subseteq L$，使得 K 的每个有效子语言都是可诊断的。寻找 G 的（部分能观）分散监控器$S_p = (\{S_i\}, \{\varphi_i\})$，使得$L(S_p/G)=L^{dact}$，并满足下面条件：

（1）$L^{dact} \subseteq K$；

（2）L^{dact}可诊断；

（3）L^{dact}尽可能大。

解决这个问题分两步：（1）从 L 得到 K；（2）从 K 得到一个关于$L^{dact} \subseteq K$能控、关于$(L, \{P_i\})$联合正态、可诊断的子语言。第一步过程可以由【53】中的方法得到。而第二步则通过将 3.3 中提供的算法加以改造便可使用。改进后的方法如下所示：

初始化：

0-1：得到 K 的产生器G^{legal}，即$L(G^{legal}) = K$。

0-2：i=0；$H_d(0) = H_d^{live}(0) = G_d^{legal}$；$M_d(0) = L(H_d(0))$；$M(0) = K$。

迭代：

模块 A：

A-1：计算$M'(i) = i\sup CUN(M(i))$；令$L(H'(i)) = M'(i)$。

A-2：构造诊断器$H'(i)$的诊断器$H_d'(i)$。

模块 B：

B-1：令 k=0；$\tilde{H}(0) = H'(i); \tilde{M}(0) = L(\tilde{H}(0)) = M'(i); \tilde{H}_d(0) = H_d'(i)$。

B-2：用 3.3.2 中的方法将$\tilde{M}(k)$扩展成有效语言$\tilde{M}^{live}(k)$。令$L(\tilde{H}^{live}(k)) = \tilde{M}^{live}(k)$，用$\tilde{H}_d^{live}(k)$表示$\tilde{H}^{live}(k)$的诊断器。

B-3：从 $\tilde{H}_d(k)$ 中除去所有关于 $(\tilde{H}_d(k), \tilde{H}_d^{live}(k))$ 对的 Stop 不定（Stop-indeterminate）状态，其结果用 $\tilde{H}_d(k+1)$ 表示。

B-4: 如果 $\tilde{H}_d(k)=\tilde{H}_d(k+1)$，则令 $H_d(i+1)=\tilde{H}_d(k)$。令 $M_d(i+1)=L(H_d(i+1))$，$H_d^{live}(k+1)=\tilde{H}_d^{live}(k)$，转到 C-1。否则令 $\tilde{M}(k+1)=P_L^{-1}(L(\tilde{H}_d(k+1)))$，且令 $\tilde{H}(k+1)$ 为 $\tilde{M}(k+1)$ 的产生器（即 $\tilde{M}(k+1)=L(\tilde{H}(k+1))$），令 k=k+1，转到 B-1。

模块 C:

C-1：如果 $H_d(i+1)=H_d(i)$，则结束。ADP 的结果是 $L^{act}=M'(i)$，否则令 $M(i+1)=P_L^{-1}(H_d(i+1))$，转到 A-1。

定理 4.3: 该方法的迭代过程可以在有限步内收敛。收敛结果 M(i)是 K 的能控、联合正态、可诊断子语言。

证明：由于该方法与 3.3 中方法结构大体相似，所以证明过程根据定理 3.3 的过程进行。根据第三章 3.3 节中的论述，从过程中 0-1 到 A-2 可以得到：

$M'(0)=i\sup CUN(M(0))$ 为 K 的能控、联合正态、可诊断子语言。

而根据推论 2.3，$M'(0)$ 是正态的。该方法中的模块 B 与 3.3.3 方法中的模块 B 大致相同，所以由【53】可以得到模块 B 的计算过程保持语言的正态性质，而且经过模块 B 处理后语言是可以诊断的，所以 $P_L^{-1}(L(H_d(1))$ 是 $M'(0)$ 的正态和可诊断子语言上界，但它不一定能控，也不一定联合正态。如果 $P_L^{-1}(L(H_d(1))$ 不满足能控和联合正态，则从 A-1 开始继续迭代，用 $P_L^{-1}(L(H_d(1))$ 代替 K，$H_d(1)$ 代替 $H_d(0)$。

根据【53】，有 $H_d(i+1)\subseteq H_d(i), \forall i>0$。而 $H_d(0)$ 是确定性状态自动机，所以该方法可在有限步内收敛。设该方法最后得到的语言是 M(i)，则它是正规的，因为它由确定性自动机 H(i)产生。而由该过程可得，M(i)是 K 的可诊断、能控、联合正态子语言得证。

如果 M(i)为所求 L^{dact}，则按 3.1.2 中分散监控器的构造方法来生成闭环语言 M(i)的分散监控器 S_P 而闭环系统的在线诊断可以由 H(i)的有效诊断器 $H_d^{live}(i)$ 来

执行，这将在下一节进行讨论。

4.3.3 在线诊断

在 4.3.2 中提供了离线诊断的方法，生成了被控的闭环语言 L^{dact}，而 L^{dact} 是在线诊断的对象。要实现分散诊断，可以采用了 4.2 中提供的方法，唯一不同的就是所诊断的对象由整个系统 G 变成了被控的闭环语言 L^{dact}。由于 L^{dact} 不一定是有效的，根据 3.3.1 中的讨论，可以通过对扩展的有效语言 $L^{dact\,live}$ 的诊断来实现。设自动机结构 G_1，有 $L(G_1)= L^{dact}$，G_1 的诊断器为 G_{1d}，而设生成相应的扩展有效语言的自动机为 G_1^{live}，相应诊断器即 G_1 的有效诊断器为 G_{1d}^{live}。则有下面定理：

定理 4.4：在协议 1 下通过对 G_1^{live} 进行分散诊断就可实现对 G_1 的诊断。

证明：根据属性 4.3，采用协议 1 对 G_1^{live} 的分散诊断的性能与集中式诊断器 G_{1d}^{live} 相同的诊断性能。而由 3.3.2 的讨论得知，对某语言的在线诊断可以通过其有效诊断器实现，即对 G_1 的诊断可以通过 G_{1d}^{live} 进行。所以对 G_1 的诊断可以采用协议 1 下对 G_1^{live} 进行分散诊断实现。得证。

进行分散诊断的各个站点分布与分散监控的站点分布相同，即每个站点都有一个监控器和扩展诊断器。设每个站点的扩展诊断器为 G_{1di}^{live}，则有 $L(G_{1di}^{live}) = P_i(L^{dactlive})$，根据扩展有效语言的构造方法可得，除去 stop 事件，本地诊断器生成的语言与本地监控器中的观测器 S_i 生成的语言相同。所以可以用本地监控器中的观测器 S_i 给本地扩展诊断器提供当前可观事件信息，从而进行本地监控。由于分散诊断与 3.3 中相比只是诊断目标不同，在满足 A1~A8 的前提下，协议 1 中任何规则都不变。

下面介绍在线诊断的系统构架。如图 4.4 所示，不失一般性，以两个站点为例。每个站点各有一个监控器 (S_i, φ_i) 和一个扩展诊断器，各个站点联合监控的结果相当于一个集中监控器 $(S, \varphi) = (\{S_i\}, \{\varphi_i\})$，使得 $L(S/G)=L^{dact}$，生成方法见 3.3。本地扩展诊断器根据 S_i 提供当前对本地可观的事件来进行本地诊断。然后每个本地诊断器根据协议 1 中的通信规则将其诊断信息以 CRi 发往协调器，与之

通信。协调器则根据协议 1 中的决定规则将各个站点发来的诊断信息综合起来，推断出故障的发生。如果协调器检测到故障，就通知故障发现模块对其进行处理。总之就是用分散监控器将系统行为限制在可诊断语言 L^{dact}，然后用分散诊断方法对 L^{dact} 进行诊断。关于采用协议 1 进行分散诊断的过程可见 4.2。

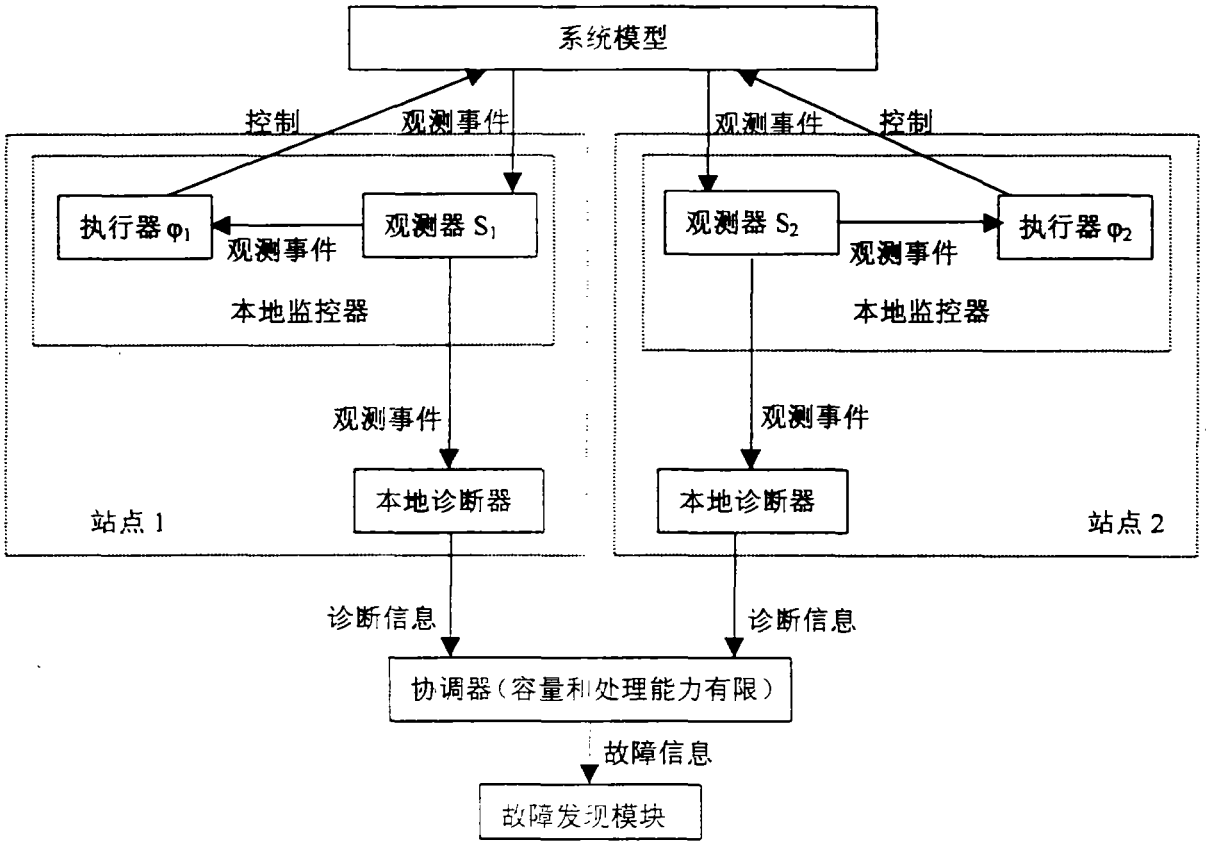


图 4.4　信息分散系统活跃诊断的系统构架

4.3.4 例子

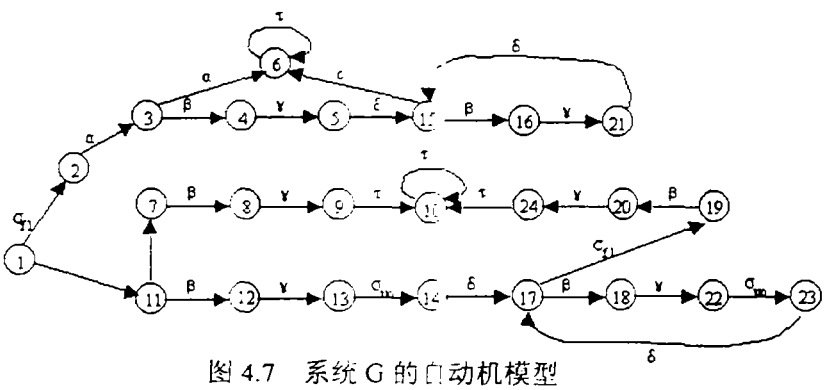


图 4.7　系统 G 的自动机模型

图 4.7 给出了一个待处理的系统 G，定义一系列事件集合：故障事件集合 $\Sigma_f=\Sigma_{f1}=\{\sigma_{f1}\}$，就一种故障类型：$F_1$。可观事件集合 $\Sigma_o=\{\alpha, \beta, \gamma, \delta, \tau\}$；可控事件集合 $\Sigma_c=\{\alpha, \beta, \gamma, \tau\}$，对事件集合 Σ 划分 $I=\{1,2\}$，对应每个本地站点。局部可观事件集 $\Sigma_{1o}=\{\alpha, \beta, \tau\}$，$\Sigma_{2o}=\{\gamma, \delta, \tau\}$。局部可控事件集合 $\Sigma_{1c}=\{\alpha, \beta, \tau\}$，$\Sigma_{2c}=\{\gamma, \tau\}$。根据上面的论述，求解过程分三个步骤：

（1） 采用 4.3.2 中的方法求出满足能控，联合正态，可诊断的子语言 L^{dact}，如图 4.6 所示。

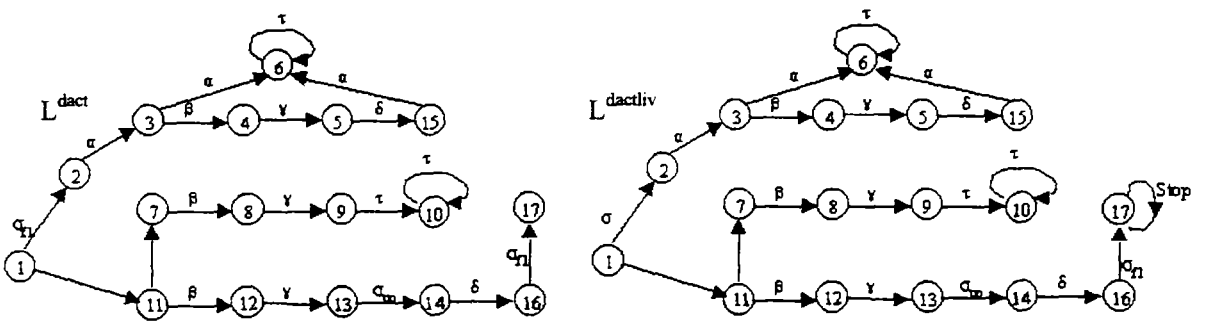
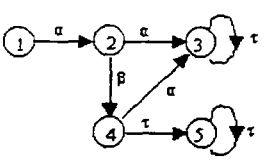


图 4.6 L^{dact} 与 $L^{dactlive}$ 的自动机模型

（2） 根据求出各个本地站点的监控器(S_1, φ_1)和(S_2, φ_2)（图 4.7）和有效语言的扩展诊断器G^e_{d1}, G^e_{d2}(图 4.8)，分散监控器的求法可见 3.2；扩展诊断器的求法可见 4.2.4，详细内容可见文献【7】。

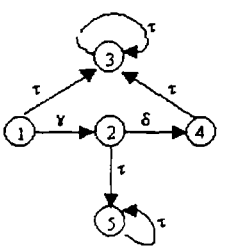
观测器 S_1：



执行器 φ_1：

状态 \ 事件	α	β	γ	δ	τ	σ_{f1}	σ_{uo}
1	1	0	1	1	0	1	1
2	1	1	1	1	0	1	1
3	0	0	1	1	1	1	1
4	0	0	1	1	0	1	1
5	0	0	1	1	1	1	1

观测器 S_2：



执行器 φ_2：

状态 \ 事件	α	β	γ	δ	τ	σ_{f1}	σ_{uo}
1	1	1	0	1	0	1	1
2	1	1	0	1	1	1	1
3	1	1	0	1	1	1	1
4	1	1	0	1	0	1	1
5	1	1	0	1	1	1	1

图 4.7 各个本地站点的监控器

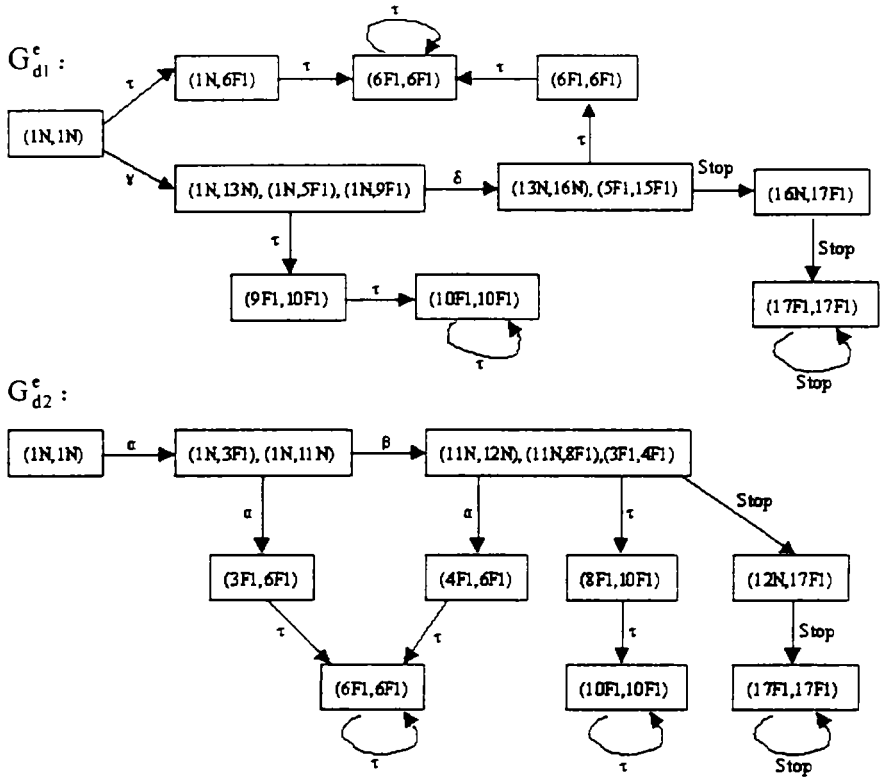


图 4.8 各个本地站点的扩展诊断器

（3）有了监控器和扩展诊断器，就可以采用图 4.4 的结构，运用协议 1 对经过分散监控限制的系统行为进行分散诊断了。

4.4 小结

本章的内容是关于一类常见的分布式系统，即信息分散系统的诊断。本章首先介绍了 Deouk 提出的协调分布式故障诊断方法，该方法是针对信息分散系统的。然后，结合上一章论述的内容，提出了对不可诊断的信息分散系统进行诊断的方法，即分散式活跃诊断方法。这种方法以 Sampath 的诊断理论为基础，将 Debouk 的分散诊断方法与 Sampath 的活跃诊断方法结合起来。该方法先通过分散监控器将系统行为限制在可诊断的范围之内，然后对闭环语言进行分散诊断。所以，该方法可以看作是活跃诊断方法在处理分散系统方面的扩展，也可以看成将 Debouk 的分散诊断方法运用于本身不可诊断的系统下。分散式活跃诊断方法由离线诊断和在线诊断两部分组成：离线诊断部分生成系统的一个能控、联合能观、可诊断的子语言，并分别建立用于生成该语言的分散监控器和诊断该语言的

本地诊断器；在线诊断部分运用了 Debouk 方法中的体系结构，采用协议 1 对目标语言进行分散诊断。

虽然前面关于分散式活跃诊断方法的讨论中只使用了协议 1，不过从 4.3.3 中讨论得知，Debouk 的诊断方法不需要经过太大改动便可直接应用于分散式活跃诊断方法的在线诊断部分，所以在线诊断也可以用协议 2 和协议 3。此外，在 4.2.5 的讨论中所使用的改进方法也可以应用于分散式活跃诊断中。当然由于协议 2 与协议 3 要在一定的条件下才能进行，所以不利于在现实中运用。

在前面的讨论中可知，分散监控器的控制目标只是使系统可诊断。不过该方法还可以适用于同时具有其他控制目标的情况。只要求出在其他控制目标作用下系统的子语言，并以该子语言作为要处理的系统，仍然可以运用分散式活跃诊断方法对其同时进行监控和诊断。

当然，这种方法还存在一些不足，比如说不能得出能控、联合正态子语言上界，而是次优解，这就导致过多限制系统行为，使其在实际中的应用受到影响。这些在今后研究中应该会得到解决。

第五章 一类分布式系统的模块化诊断方法

5.1 引言

基于离散事件动态系统的诊断方法已经运用于各类复杂系统的诊断中，而在各类复杂系统中，最常见的就是分布式系统。对于分布式系统的诊断问题引起了广泛的关注[65]~[70]。

我们将要研究的一类分布式系统具有下面两个方面特征：

（1） 系统由许多组件构成，且存在共享资源。系统的各个组件由于需要访问共享资源，所以会与其他组件发生相互作用。

（2） 组件的数量和位置不是一成不变的，增加和减少组件比较频繁。

很明显，计算机和通信网络，制造业，过程控制，远程控制多机器人系统和电力系统等大规模分布式系统就具备上述特征。

由于上述特征，如果我们采取传统的集中式诊断方法，就得重构整个系统行为，并构造整个系统的诊断器。由于这类系统比较庞大和复杂，整个构造过程费时费力；每当加入或者去除一个组件时，就得重新构建一切，而且这种改变是经常的，这极大地增加了开销。为了解决这个问题，我们可以针对每个组件建立一个本地诊断器，而不用考虑整个系统行为。这样当增加或去除一个组件的时候，和它无相互作用的组件的本地诊断器就不受任何影响。这种模块化诊断的思想便能较好的解决上面提出的问题。当然，采用模块化诊断的前提是没有属于整个系统范围的故障需要诊断。

文献【69】中采取了上述的模块化诊断思想来解决这类系统的诊断问题，但它采用的方法有着下面的前提条件：当系统中发生故障时，会产生可见的告警，且一个告警对应着几个不同的故障的发生。但在现实中，故障的发生不一定存在可见的告警，所以【69】中的方法在应用中存在着不足。基于这种考虑，我们在这种模块化诊断思想的基础上运用了诊断器理论（见第三章 3.2），提出了一种新的模块化诊断方法，以适应上述情况。而本章中，先介绍模块化诊断的思想，然后再详细叙述这种基于诊断器理论的模块化诊断方法。

5.2 模块化诊断

我们所研究的分布式系统由 n 个组件构成，每个组件 i 的行为都可以用自动机模型 $C_i=(X_i,\Sigma_i,\delta_i,x_{i0})$ 表示(i =1,2,...,n)；其中 X_i 表示状态集合，Σ_i 表示事件的集合，$\delta_i:\Sigma_i\times X_i\to 2^{X_i}$ 为转移函数，x_{i0} 表示初始状态。假定一个组件无法观察到其他组件中发生的事件，即 $\Sigma_i\cap\Sigma_j=\phi$ $(i,j\in\{1,2,...n\}且i\neq j)$。

在文献【69】中，故障是在状态中发生的，用 $X_{if}(\subseteq X_i)$表示故障状态的集合。报警可以表示成从故障状态出发的转移，所以报警即 Σ_i 中一些事件。

组件的事件由两种类型：$\Sigma_i=\Sigma_{i,g}\cup\Sigma_{i,l},\Sigma_{i,g}\cap\Sigma_{i,l}=\phi$；其中 $\Sigma_{i,g}$ 表示与访问共有资源有关的事件集合，即全局事件，$\Sigma_{i,l}$ 则表示本地事件的集合。全局事件是因为存取共享资源而发生的事件。在这类系统中，一个组件总是与其他组件发生相互作用。我们可以用一个自动机模型来建模 k（$0\le k\le n$）个组件间的相互作用的行为，而该自动机模型的事件集合等于这 k 个组件的全局事件集合的并。为了使用模块化诊断的方法，我们引入了一个全局行为的概念：组件 i 的全局行为就是指将 C_i 和涉及组件 i 的所有相互作用的行为进行||运算得到的结果（||运算的定义可见文献【11】）。下面以 k=2 的情况为例说明之：

当 k=2 时，设组件 i 和 j 之间有相互作用，其行为可以用自动机模型 $C_{ij}=(X_{ij},\Sigma_{i,g}\cup\Sigma_{j,g},\delta_{ij},x_{0,ij})$表示，其中 i, $j\in\{1,2,...n\}$且$i\neq j$，$X_{ij,f}(\subseteq X_{ij})$表示故障状态的集合。如果组件 i 只与组件 j 有相互作用，则组件 i 的全局行为等于 $C_i\|C_{ij}$。

要构造诊断器，首先构造观测器。我们用 $obs(C_i)=(Q,\Sigma_{i,o},\delta_{obsi},q_0)$表示 $C_i\|C_{ij}$ 的观测器，其中 $\Sigma_{i,o}$ 是 Σ_i 中可观事件的集合，Q 中的状态形式为$\{(x,x')|x\in X_i且x'\in X_{ij}\}$。

这里诊断器的构造方法与 3.2 有所不同，故障标识的集合为{Y, N}, N 表示状态正常，而 Y 则表示有故障发生。用 $Diag(C_i)=(D_i,\Sigma_{i,o},\delta_{diagi},d_0)$表示基于 $C_i\|C_{ij}$

的诊断器。D_i 中每个状态 d 都对应与相应的观测器中的一个状态 q，即 d={(x，x'，l) | ((x,x')∈q，l∈{Y, N})。Diag(C_i)构造方法见文献【69】，由于与本文关系不大，在此省略。

5.3 基于诊断器理论的模块化诊断方法

5.3.1 系统模型

如果分布式系统的故障事件是不可观的，且故障发生后无法观测到报警，那么文献【69】中的模块化诊断方法就不能直接运用了。为此我们对这种方法进行了如下的改进：

假设系统由 n 个组件构成，系统的模型与 2 中介绍基本相同，只是没有故障状态这个概念。在我们要处理的系统中，在每个组件 i 中，都有故障事件集 $\Sigma_{i,f} \subseteq \Sigma_i$，且故障事件都是不可观的，故障事件集根据不同的故障类型被分成 m 个互不相交的非空集合：$\Sigma_{i,f} = \Sigma_{i,f1} \cup \Sigma_{i,f2} \cup \ldots \cup \Sigma_{i,fm}$，这种故障划分用 $\Pi_{i,f}$ 表示。假设故障事件必定属于系统中的一个组件，所以整个系统中所有故障事件集合 $\Sigma_f = \Sigma_{1,f} \cup \Sigma_{2,f} \cup \ldots \cup \Sigma_{n,f}$。通过对每个组件的故障事件集合的划分，我们可以对整个系统的故障进行分类，这种划分用 Π_f 表示。在 Π_f 作用下，$\Sigma_f = \bigcup_i (\Sigma_{i,f1} \cup \Sigma_{i,f2} \cup \ldots \cup \Sigma_{i,fm_i})$ (i=1,2,...n)，其中 $\Sigma_{i,f} = \Sigma_{i,f1} \cup \Sigma_{i,f2} \cup \ldots \cup \Sigma_{i,fm_i}$ 是使用划分 $\Pi_{i,f}$ 的结果。

5.3.2 模块化诊断器的构造

考虑组件 i 的行为 $C_i = (X_i, \Sigma_i, \delta_i, x_{i0})$，并给定相应的可观事件集 $\Sigma_{i,o} \subseteq \Sigma_i$，故障事件集 $\Sigma_{i,f}$，$\Sigma_{i,f} \subseteq \Sigma_i - \Sigma_{i,o}$。用 G_i 来表示组件 i 的全局行为，由 2 中可知，G_i 的事件集 $\Sigma_{ag,i} = \Sigma_i \cup \Sigma_{k_1,g} \cup \Sigma_{k_2,g} \cup \ldots \cup \Sigma_{k_p,g}$，其中 $\{k_s\}$(s=1,2,...,p)为与组件 i 有相互作用的组件的集合，而 $\Sigma_{ks,g}$ 为组件 k_s 的全局事件集(s=1,2,...,p)。这样，在可观映射 $P_{i,g}: \Sigma_{ag,i} \to \Sigma_{i,o}$，故障事件集 $\Sigma_{i,f}$ 及基于 $\Sigma_{i,f}$ 的故障分类 $\Pi_{i,f}$ 的条件下就可以

构造针对 G_i 的诊断器 $Diag(G_i)$（构造方法可见第三章 3.2.4）。这样得到的诊断器 $Diag(G_i)$就是我们要构造的组件 i 的模块化诊断器。

为了说明上述构造模块化诊断器的思想，下面举例说明其构造过程：

例 5.1 组件 i 的行为 $C_i=(X_i,\Sigma_i,\delta_i,x_{i0})$，$\Sigma_{i,g}=\{a, b, c\}$，$\Sigma_{i,l}=\{\}$，$\Sigma_{i,o}=\Sigma_{i,f}=\Sigma_{i,fl}=\{b\}$自动机模型见图5.1，$x_{i0}=0$；与i相互作用组件为j，$\Sigma_{j,g}=\{d, e\}$；相互作用的行为 $C_{ij}=(X_{ij},\Sigma_{i,g}\cup\Sigma_{j,g},\delta_{ij},x_{0,ij})$，自动机模型见图5.2，$x_{0,ij}=0$。则组件i的全局行为 $G_i=C_i\|C_{ij}$，自动机模型见图5.3；由 G_i 生成组件i的模块化诊断器 $Diag(G_i)$，如图5.4所示。

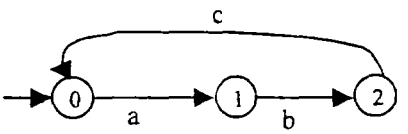


图 5.1 组件 i 的自动机模型 C_i

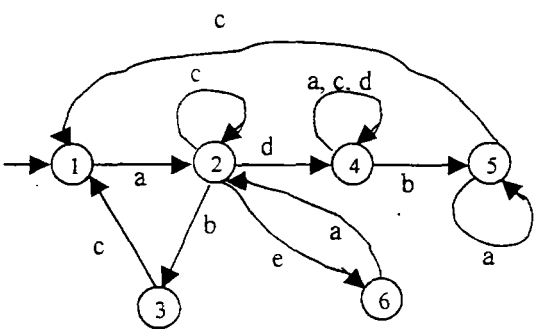


图 5.2 组件 i 与组件 j 的相互作用行为的自动机模型 C_{ij}

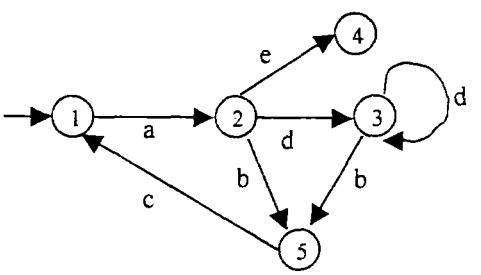


图 5.3 组件 i 的全局行为的自动机模型 G_i

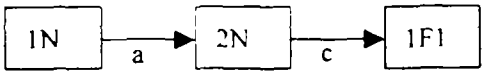


图 5.4 组件 i 的模块化诊断器 Diag(G_i)

5.3.3 诊断条件

考虑到分布式系统的特点，我们根据文献【49】中的讨论修改了定义 3.6 中的一般性可诊断条件，重新定义如下：

定义 5.1： 闭语言 L 关于可观映射 P 和在故障集合 Σ_f 上的划分 Π_f 是可诊断的，如果满足下面条件：

$$(\forall i \in \Pi_f)(\exists n_i \in N)(\forall s \in \Psi(\Sigma_{fi}))(\forall t \in L/s)[((\|P(t)\| \le n_i) \wedge (L/st = \phi) \Rightarrow D_1) \wedge (\|t\| \ge n_i) \Rightarrow D_2)$$

其中 $D_1:(\omega \in P_L^{-1}(P(st)) \wedge (L/\omega = \phi) \Rightarrow \Sigma_{fi} \in \omega;\ D_2 : \omega \in P_L^{-1}(P(st)) \Rightarrow \Sigma$

在该定义中，被诊断系统不需要满足 3.2.1 中的两个假设条件。由于本文中只要求在故障发生后，再发生有限个可观事件，即可探测出故障的发生；而根据文献【49】和【53】中定义，要求在故障发生后，再发生有限个事件，便可探测出故障。这种修改使得被诊断系统不用满足条件（2）。例 5.1 中的模块化诊断器就符合定义 5.1 的诊断条件，所以是可诊断的。

根据定义 5.1，可以得到不同于常规集中式系统诊断条件的分布式系统模块化诊断条件：

定理 5.1： 对于上面所讨论的分布式系统，如果故障必定属于其中一个组件，且每个组件 i 的全局行为 G_i 都满足定义 5.1 中的诊断条件，则通过模块化诊断器所得到的诊断信息不少于在基于故障集合 Σ_f 的分类 Π_f 下，对系统 G 进行集中式诊断所得到的信息。

【证】对系统 G，$G=\|_i G_i$ (i=1, 2 ....n)，定义映射 $P_i:\Sigma \to \Sigma_{ag,i}$，其中 $\Sigma=\bigcup_i \Sigma_i$ (i=1, 2, ... n)。设 G 中任意一个含故障的事件序列 $s \in L(G)$，则对应于每个组件 i，$P_i(s) \in L(G_i)$。而 s 中每个故障必属于一个组件的事件集。取 s 中任意故障 σ，不失一般性，设 $\sigma \in \Sigma_{i,fk} \subseteq \Sigma_{i,f} \subseteq \Sigma_i$。根据定理 5.1 的条件，$G_i$ 是可诊断的，则模块化诊

断器 Diag(G_i)根据 $P_{i,g}(P_i(s))$ 便可以诊断出 $P_i(s)$中发生了 $\Sigma_{i,fk}$ 类故障。而因为 $\Sigma_i \cap \Sigma_j = \phi$ （j≠i 且 j=1,2,...n），所以 $\sigma \notin \Sigma_j$ （j≠i 且 j=1,2,...n）。综上所述，系统 G 中不管发生了哪类故障，都有相应组件的模块化诊断器诊断出故障的发生，而其他组件的故障诊断器对不属于自己事件集合的故障不予检测。所以在满足定理 5.1 的前提条件的情况下，所有组件的模块化诊断器一起便能诊断出所有的故障类型，即能诊断出 Π_f 中的每一类故障。由此可见模块化诊断器所能得到的诊断信息不少于对系统进行集中式诊断的情况。得证。

5.3.4 在线诊断

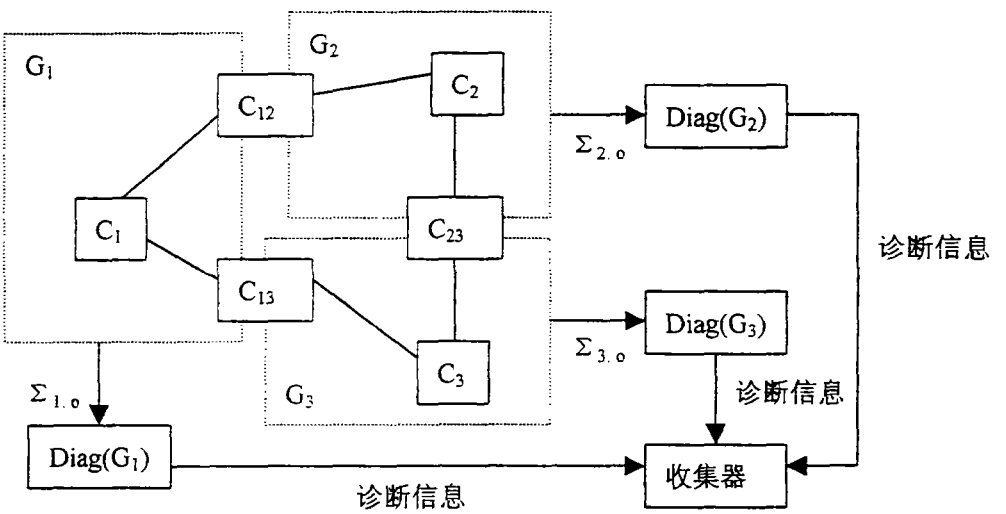


图 5.5 分布式系统的模块化诊断原理图

对于一个上述的分布式系统，在每个组件上配备一个模块化诊断器，以分别诊断各个组件；同时还给整个系统配备了一个收集器，它负责收集各个组件的模块化诊断器所得到的诊断信息。图 5.5 给出了在线诊断的原理图。不失一般性，图中 G 由 3 个组件构成，且只考虑只有两个组件相互作用的情况，其中 $G_1 = C_1 \| C_{12} \| C_{13}$，$G_1 = C_2 \| C_{12} \| C_{23}$，$G_3 = C_3 \| C_{13} \| C_{23}$。在分布式系统中，每个模块化诊断器从初始状态开始就与组件同步运行。在满足定理 5.1 的情况下，系统任何一个模块发生了故障，便按照下面的过程进行诊断和处理：

（1） 相应的诊断器在有限个可观事件发生后，由诊断器中的相应状态上所附加的故障信息判断出故障发生及故障类别，其诊断方法见第三章的

3.2.6。

（2） 诊断器诊断后，将诊断信息和发生故障的组件代号发给收集器。

（3） 收集器接收到发自诊断器的诊断信息后，判别出故障发生的模块、故障类别；有关人员根据这些信息采取相应的措施排除故障。

5.4 小结

针对网状结构的分布式系统，本章提出了一种模块化诊断方法来解决其诊断问题。该方法为每个组件构造诊断器，即模块化诊断器。每个模块化诊断器独立进行诊断，独立地提供诊断信息。这种分而治之的方法很适应这类系统的分布式和结构变化大的特点，能很好的完成诊断任务。此外，这些模块化诊断器的构造工作可以同时进行，这就大大减少了开销。

当然，要使该方法能有效进行诊断，需满足一定的条件：（1）组件之间无共同的事件；（2）不存在整个系统范围的故障事件；（3）每个组件的全局行为都是可诊断的。对于前两个条件，可以在建模的时候尽量满足。而第三个条件可以通过改进诊断方法，即将 3.3 中讨论的活跃诊断引入到这类分布式系统的诊断中来实现。这种活跃诊断的思想已经用于信息分散系统的故障诊断中了（见第四章 4.3），至于它在这类分布式系统的诊断中的应用，还有待进一步的研究。

第六章 DEDS 故障诊断理论的应用

6.1 简介

DEDS 故障诊断方法已经被广泛应用与各类复杂系统的诊断中，主要是下面这些领域：

（1）工业上的一些实时系统[75, 77, 84, 88, 90, 91]；

（2）电力系统[66, 67, 68]；

（3）通讯网络[74]和电信网络[79, 80, 82, 87]；

（4）智能公路系统[78, 83, 85, 89]；

（5）自动化制造系统[73, 77]；

（6）集成电路制造[76]和数字电路分析[81]；

（7）间歇生产过程控制[86]；

诊断器理论作为 DEDS 故障诊断方法的重要分支，其应用比较广泛。特别是集中式诊断方法，已经应用于供暖、通风和空调（HVAC）系统[91]，文档处理系统[90]，智能公路系统[89]等领域。密西根大学则将诊断器理论开发成软件工具，并应用于工业中的实时系统方面，下面是几个例子：

（1）John 控制公司的采暖、通风和空调系统的接线盒的诊断分析[75]；

（2）Rockwell 科技中心的传送带系统的诊断[77]；

（3）施乐公司一种数字复印机的进纸系统的诊断[84]。

本章将举出一些诊断器理论应用方面的例子并加以分析。我们首先要讨论一下如何用给定的信息为被诊断的系统建模的问题，并以例子进行分析；接下来将通过设计一个用于泵－阀门系统的诊断控制器的过程来说明活跃诊断方法的应用。在本章的最后，将给出一些结论。

6.2 使用诊断器理论进行故障诊断

6.2.1 诊断体系结构和诊断过程

图 6.1 分析了一个集中式监控系统的全局概念上的两层构架[50, 63]，该系统包含一个基于 DES 的故障诊断子系统。下层是系统本身，它还包含底层控制器；

这些控制器由设备控制器和连续变量控制器组成。上层是一个监控器，它负责控制并对下层控制器、故障诊断、故障发现进行协调。监控器还必须在故障被识别出来以后重新配置系统。两层之间的接口负责将系统中可观事件的发生通知监控器并将监控器发出的命令传达给系统。

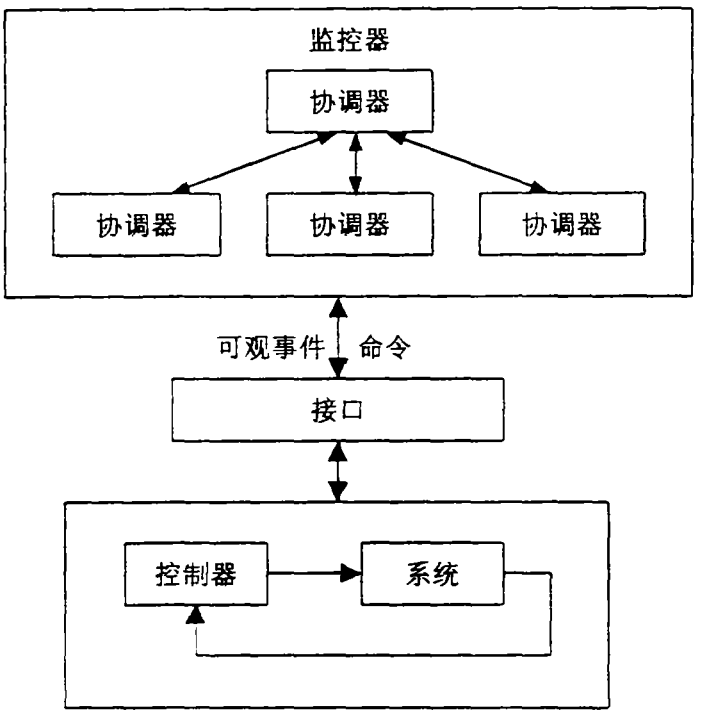
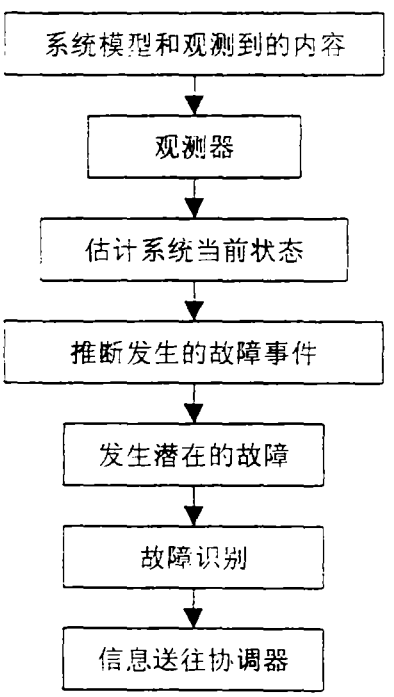


图 6.1 全局系统体系结构



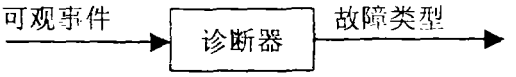


图 6.2 诊断过程

故障诊断主要分为两个步骤：先构造被诊断系统的一个离散事件模型，然后根据这个模型构造诊断器。图 6.2 给出了诊断的过程。诊断器方面的理论在第三章中已经详细介绍过，这里就不再重复了。下一节主要讨论一下系统建模的问题。

6.2.2 系统建模

上面说到，诊断的第一步就是构建系统的离散事件模型。系统通常都是由一些组件构成的，而且还配备有一系列的传感器。我们可以采用一个系统的方法，从这些组件的离散事件模型和具有离散值的这些传感器得到一个系统的整体模型，该模型是基于事件的，可以捕捉到组件间的相互作用，并能通过传感器将它们合并起来。这个整体模型就是诊断器的诊断对象。下面详细介绍之。

假设系统由 n 个组件构成，先分别对这些组件建立离散事件的自动机模型，设组件 i 的自动机模型为 $G_i = (X_i, \Sigma_i, \delta_i, x_{0i})$。令 $\tilde{G} = (\tilde{X}, \tilde{\Sigma}, \tilde{\delta}, \tilde{x}_0)$，$\tilde{G} = \|_{i=1}^{n} G_i$。$\|$ 是并行组合运算符，即第二章 2.3.1 中提到的织积。

假设系统配有 M 个传感器，每个传感器可以用映射方式表示，即 $h_j : \tilde{X} \to Y_j, j = 1, 2, \cdots, m$。其中 Y_j 表示第 j 个传感器可能的输出离散值集合。可以定义 $Y = \prod_{i=1}^{m} Y_j$，相应的可以定义映射 $h : \tilde{X} \to Y$，对于任意状态 $x \in \tilde{X}$，有：

$h(x) = (h_1(x), h_2(x), \cdots, h_m(x))$。

最后，我们重新定义 $\tilde{G}$ 中的状态迁移函数，以得到作为诊断对象的整体模型 $G = (X, \Sigma, \delta, x_0)$，其中 $x_0 = \tilde{x}_0$。下面给出从 $\tilde{\delta}$ 生成 δ 的步骤：

先假设 $\tilde{\delta}$ 中的一个转移为 $\tilde{\delta}(\sigma, x) = x'$，其中 $x, x' \in \tilde{X}$，$\sigma \in \tilde{\Sigma}$，然后根据下面的过程来生成 δ 中的转移：

（1）如果 σ 可观，则将 $<\sigma, h(x)>$ 定义为 Σ 中的可观事件，并令 $\delta(<\sigma, h(x)>, x) = x'$。

（2）如果σ不可观，且有 $h(x)=h(x')$，则定义σ为Σ中的不可观事件，并令 $\delta(\sigma,x)=x'$。

（3）如果σ不可观，且有 $h(x)\neq h(x')$，则增加一个新状态 x_{new}，并有下面两个转移：$\delta(\sigma,x)=x_{new}$，$\delta(<h(x)\rightarrow h(x')>,x_{new})=x'$。同时在Σ中增加一个可观事件 $<h(x)\rightarrow h(x')>$，表示从状态 x 变化到状态 x′过程中传感器值的变化。

根据上面的过程，就可以得到 G 中的状态迁移函数δ和事件集合Σ。由上面的方法可见Σ中包含三种事件：

（1）$<\sigma,h(x)>$：可观事件；

（2）σ：不可观事件；

（3）$<h(x)\rightarrow h(x')>$：可观事件。

我们用 X_{new} 表示通过上面 3 个步骤所生成的所有新的状态 x_{new} 的集合，所以 G 中的状态集 $X=\tilde{X}\cup X_{new}$。这样就完成了构造整个系统复合模型的过程。我们就得到了一个可以用于诊断的系统完整模型 $G=(X,\Sigma,\delta.x_0)$。G 中同时包含了系统中正常时和发生故障时的行为。系统中的可观事件有两种：（1）监控器发出的指令和执行指令后瞬间传感器的读数；（2）传感器读数的变化。系统中的不可观事件则是故障事件或者其他导致系统状态变化却没有被传感器记录的事件。

当然，这种建模方法不局限于发生装置故障和控制器故障的情况，传感器故障也可以用这种框架处理。如果存在传感器故障，那么只要将传感器看成系统的附加的组件，同样用这种方法来生成系统模型。

6.2.3 例子与分析

为了更好的说明这种方法，我们将举出一个例子并加以分析说明。

我们可以构造一个机械系统，它由泵、阀门和控制器构成。假设泵和控制器都没有故障，而阀门有两个故障，即阀门开时卡住（stuck-open）和阀门闭时卡住（stuck-closed）。而且只有在阀门是开启的时候才会发生 STUCK_OPEN 类故障，而只有在阀门关闭的时候才会发生 STUCK_CLOSED 类故障。为了建立整个系统的自动机模型 G，我们分 3 步进行：

（1）分别建立泵、阀门和控制器的自动机模型。状态 VC, SC 分别表示阀门关闭时系统正常和不正常时的两个状态，而状态 VO. SO 分别表示阀门开启时系统正常和不正常时的两个状态。事件 OPEN_VALVE 和 CLOSE_VALVE 是可观事件，分别表示开启阀门和关闭阀门。而在泵模型中，状态 POFF 表示泵已经关闭，PON 表示泵已经打开；两个事件 START_PUMP 和 STOP_PUMP 是可观事件，分别表示打开泵和停止泵。而事件 LOAD 和 NO_LOAD 分别表示需要抽水和不需要抽水两个事件，也是可观的。这三个组件的自动机模型如图 6.3 所示：

（2）按 6.2.2 中方法用映射方式表示传感器。系统配备了一个传感器，称为

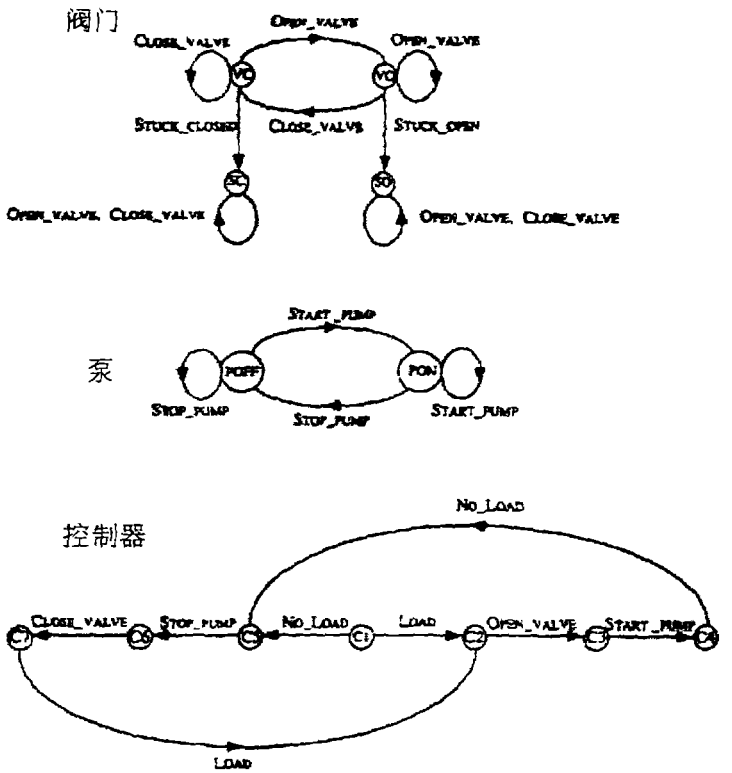


图 6.3 泵一阀门一控制器系统的各个组件的模型

水流传感器，它的输出被离散化成两个可能的值：F 和 NF。当有水流的时候输出为 F，如果没有水流，输出为 NF。映射方式表示的传感器如表 6.1 表示。表 6.1 中 • 表示控制器中的状态，说明传感器输出值与控制器中状态无关。

h(VC,POFF,•) =NF
h(VO,POFF,•) =NF
h(SC,POFF,•) =NF
h(SO,POFF,•)=NF

h(VO,PON,•) =F
h(VC,PON,•) =NF
h(SC,PON,•) =NF
h(SO,PON,•)=F

表 6.1 用映射方式表示的传感器

（3）用 6.2.2 中介绍的方法生成整个系统的自动机模型 G。如图 6.4 所示，图中为了方便起见，状态用数字符号表示，每个状态都对应一个三元组（即（阀门状态，泵状态，控制器状态）），只要根据图中对应的标识便可求出任意一个状态的三元组表示。图中的可观事件的转移用实现标识，不可观事件转移用虚线标识。

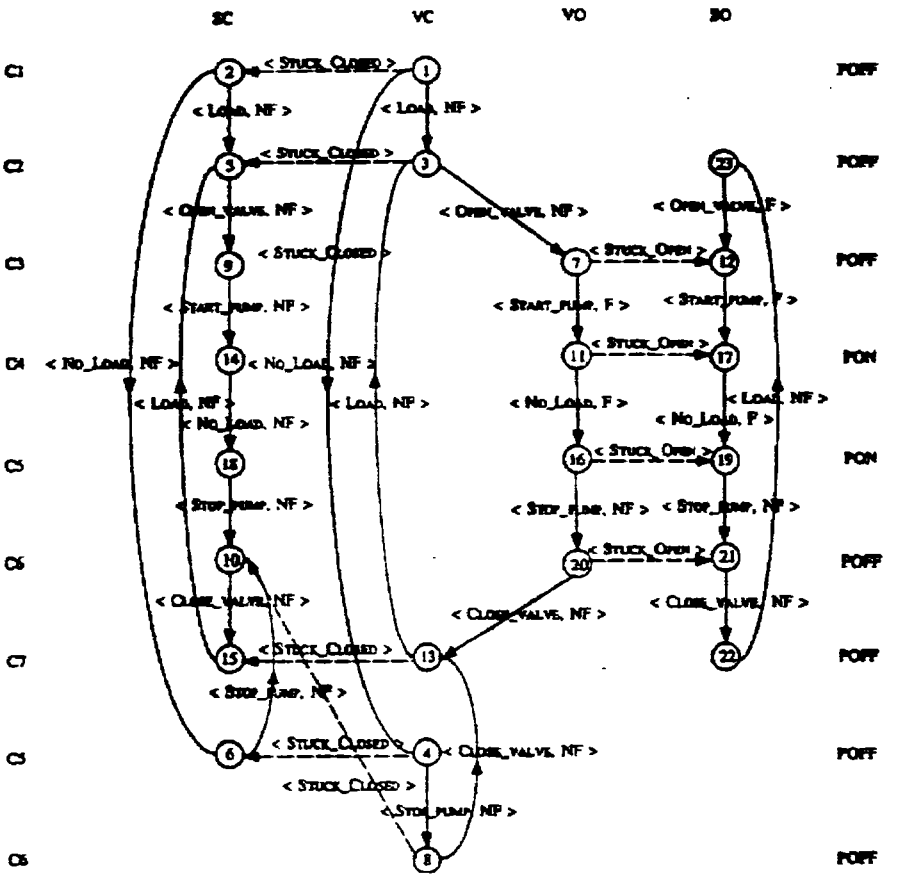


图 6.4 泵一阀门一控制器系统的整体模型 G

经过上面 3 个步骤，就得到了系统的整体模型 G。令 Σ_{f1} ={STUCK_CLOSED}，Σ_{f2} ={STUCK_OPEN}；分别表示 F1 类故障和 F2 类故障。接下来，我们根据第三章 3.2 中提供的诊断器构造方法构造系统 G 的诊断器 G_d，如图 6.5 所示。通过对 G_d 的分析，发现其存在一个 F1 不确定环，根据定理 3.1，系统是不可诊断的。由此可见，单纯的诊断器理论难以对一些系统实施诊断。而活跃诊断方法的提出则较好的解决了这一问题，使得诊断器理论能更好的应用于实际。下一节中，将举例对活跃诊断进行分析。

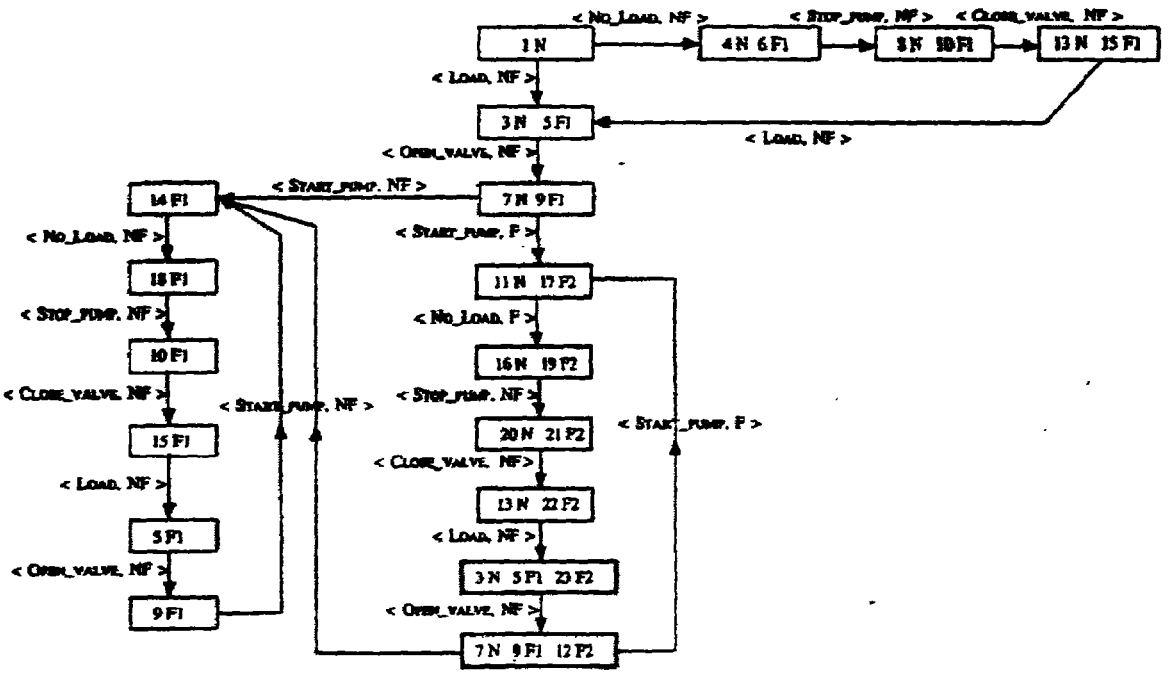


图 6.5 泵－阀门－控制器系统的诊断器 G_d

6.3 活跃诊断应用举例

在第三章 3.3 节中，我们提到了一种活跃诊断的方法，它将监控于诊断结合起来，较好的解决了不可诊断系统的诊断问题。下面，我们将通过为一个系统设计一个诊断监控器的过程来对这种方法进行分析说明。

我们采用的系统与 6.2 中的类似，即泵－阀门－控制器系统，除了控制器稍做改变外，泵和阀门的模型以及传感器等都与 6.2 中相同。控制器模型如图 6.6 所示：

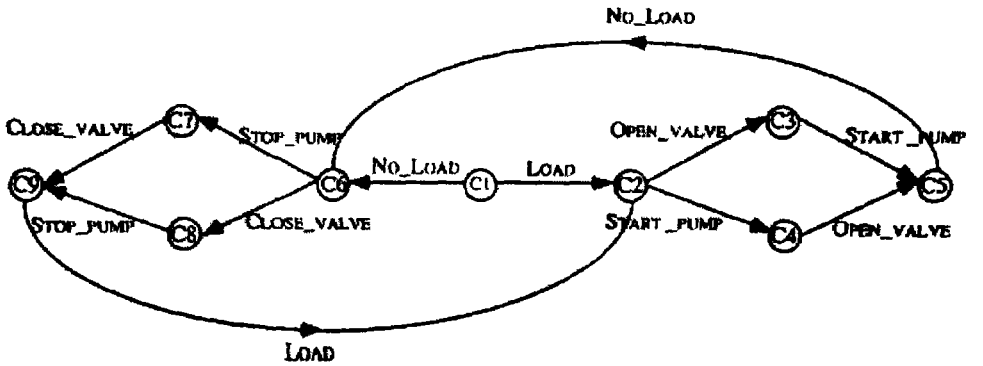


图 6.6 控制器模型

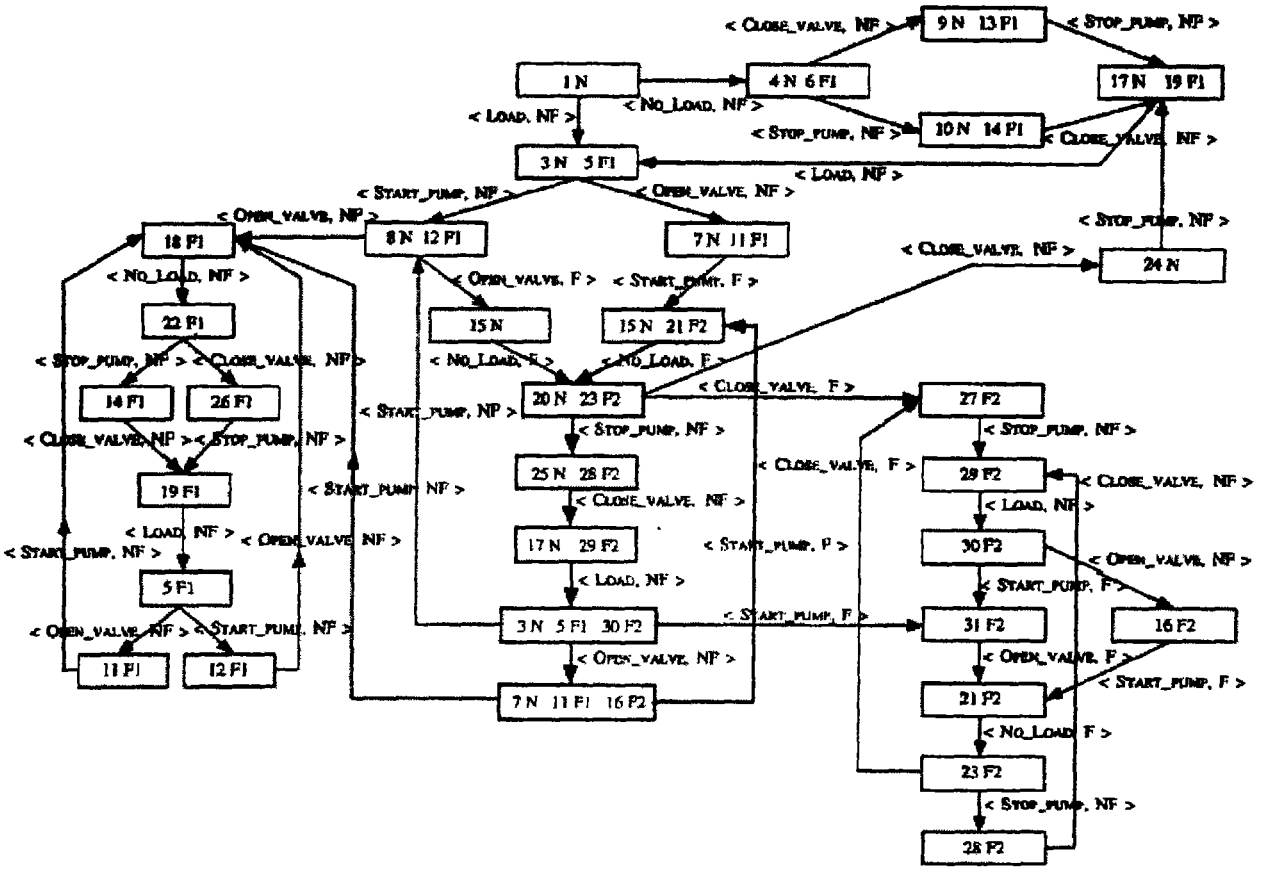


图 6.7 诊断器 G_d

同样可以用 6.2.2 中的方法构造出系统的整体模型 G，由于产生的模型太大，这里省略了。与 6.2.3 中例子不同的是，我们必须区分出可控事件集合和不可控事件集合。在 G 中，可控事件集合 Σ_c={<OPEN_VALVE，NF>, <OPEN_VALVE，F>, <CLOSE_VALVE，NF>, <CLOSE_VALVE，F>, <START_PUMP，NF>, <START_PUMP，F>, <STOP_PUMP，NF>, <STOP_PUMP，F>}，而不可控事件集合 Σ_{uc} ={<LOAD，NF>，<LOAD，F>，<NO_LOAD，NF>，<NO_LOAD，F>，

STUCK_OPEN, STUCK_CLOSE}。而故障事件和分类都与 6.2.3 相同。我们用 3.2 中的方法生成诊断器，如图 6.7 所示。由图 6.7 可以看出，诊断器存在一个 F2 不确定环，所以系统 G 是不可诊断的。

我们可以采用 3.3.3 中的方法来求出诊断控制器。经过迭代计算，我们得到了系统 G 的诊断控制器。由于整个诊断控制器模型过大，图 6.8 只给出了诊断控制器部分实现的模型。当然，这样增加了对系统行为的限制，使得系统行为除了被限制在控制目标以内外，还必须限制在可诊断范围内。

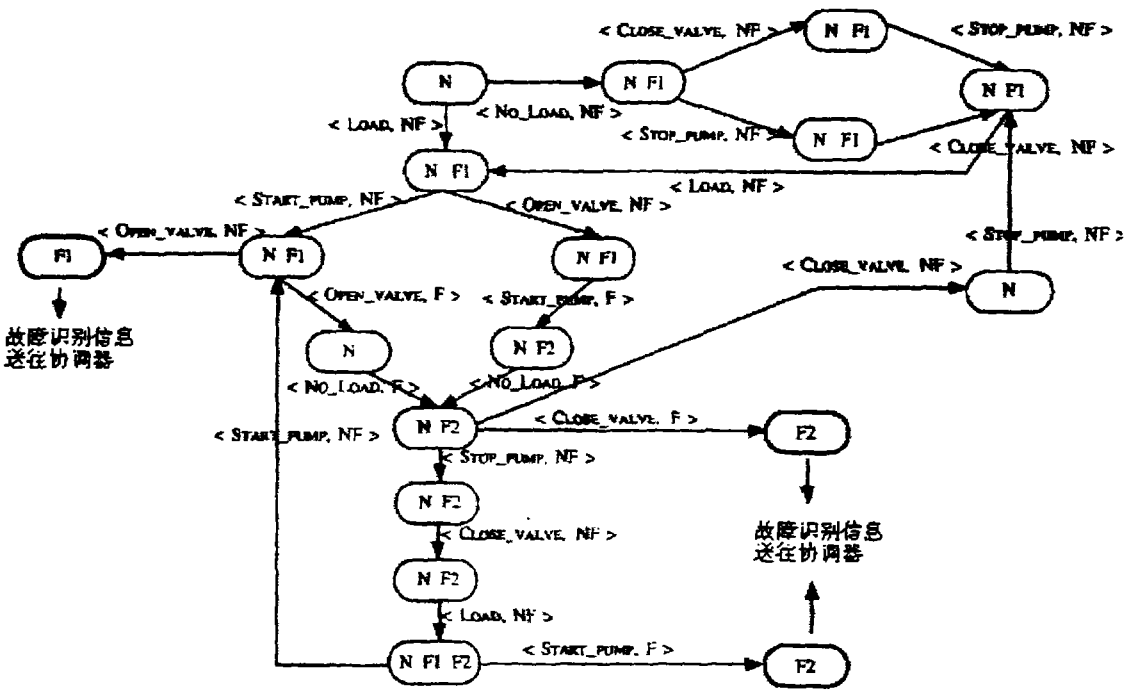


图 6.8 系统诊断控制器的部分模型

6.4 小结

本章主要讨论 DEDS 故障诊断方法的应用问题，首先简要介绍了 DEDS 故障诊断方法的应用领域和一些具体应用实例，然后就诊断器理论的应用方面的一些方法进行了讨论，包括进行故障诊断的系统的体系结构、诊断过程，以及建模方法等。另外，还通过一些例子来分析如何将诊断器理论应用于具体的系统。

第七章 结束语

随着科学技术的发展，生产系统规模日益扩大，结构日趋复杂，而且出现了许多“人造系统”，故障的发生也日益频繁。由于现代社会对系统可靠性的要求日益提高，故障诊断就显得尤为重要。但是许多系统特别是复杂系统中存在着大量具有异步性、离散化和事件驱动等特点的故障，使得传统的故障诊断方法难以有效进行。系统的复杂性以及故障的事件化特点更适合离散事件动态系统的概念来处理，而离散事件动态系统的各个理论经过较长时间的发展，已经比较成熟，所以基于离散事件动态系统的故障诊断方法就应运而生。虽然这类故障诊断方法出现不久，还有大量问题需要解决，但已经取得了丰富的成果。

在这样的背景下，作者开始了对离散事件动态系统诊断方法的研究。由于DEDS的建模方式多种多样，所以DEDS故障诊断方法也丰富多彩。我们选取了DEDS故障诊断方法中比较有影响的诊断器理论作为主攻方向，在阅读了大量关于监控理论和诊断器理论及其应用的文献的基础上，结合其他诊断方法，对DEDS故障诊断方法进行了一定的研究，取得了下面的一些主要成果：

（1） 在第一章中全面、系统的阐述了DEDS故障诊断理论的产生原因和发展过程，总结了目前DEDS故障诊断的各种常见方法，简要的介绍了这些方法的基本原理，并对DEDS故障诊断的优点和意义进行了阐述。

（2） 本文中重点讨论的DEDS故障诊断方法是来源于监控理论，所以在第二章中我们较为详细地讲述了监控理论。我们首先介绍了DEDS监控理论中的一些基本知识和在监控理论中起比较重要作用的一些子语言及其优化语言的求解方法。这些都是后面章节讨论的基础。考虑到针对复杂系统的监控是监控理论的重要研究方向，所以本章也着重讲述了分散监控方法，包括模块化监控方法以及由此引出的针对局部目标和全局目标的分散监控方法。在前面论述的基础上，作者提出了用于全局目标分散监控的联合正态理论，该理论能较好的解决分散监控的实现方面的问题，而且对DEDS故障诊断理论研究也有一定的作用，比如在后面部分提出的分散活跃诊断方法就是以联合正态理论为基础的。

（3） 诊断器理论是DEDS诊断理论中的一个重要分支，本论文的研究主要是围

绕诊断器理论进行的。所以在第三章中比较详细的介绍了诊断器理论，包括概念，模型，诊断原理等。针对不可诊断系统的活跃诊断方法则使得诊断器理论能实际中能更好的应用，本章也对活跃诊断方法进行了叙述。而这些都是基础性知识，我们将在本章的基础上提出了一些新的方法，这些将在后面章节进行介绍。

（4） 同监控理论一样，处理复杂系统一直是 DEDS 故障诊断理论的一个努力方向，而很多复杂系统都具有信息分散的特点，所以对信息分散系统的诊断就引起了广泛的关注。Debouk 提出的协调分布式故障诊断方法就很有代表性，所以在第四章中对它进行了介绍。在协调分布式方法和活跃诊断方法的基础上，我们提出了一种针对信息分散系统的分散式活跃诊断方法，使得诊断器理论能更好的应用于具有信息分散特征的系统，尤其是那些不具备可诊断性的系统。

（5） 像计算机网络之类的由相互作用的组件构成的大型分布式系统，进行集中式诊断代价太大。基于已有的一些采用模块化诊断来处理这类问题的方法，第五章提出了一种基于诊断器理论并运用模块化诊断的方法。该方法能有效处理结构变化比较大的复杂分布式系统的诊断问题，具有很好的现实意义和研究前景。

（6） DEDS 故障诊断方法不仅在理论研究上取得了较大的发展，而且已经开始广泛运用于实际的生产和生活中，所以第六章讨论了 DEDS 故障诊断方法的应用，并举了一些应用方面的例子。

（7） 在第七章中，先对全文进行总结，并讨论了 DEDS 故障诊断领域特别是监控器理论这方面所存在的一些悬而未决的问题，最后对 DEDS 故障诊断领域的研究和应用的前景进行了讨论。

DEDS 故障诊断方法是以离散事件动态系统为基础的，离散事件动态系统理论及其基础学科的发展必然会带动 DEDS 故障诊断方法的发展，而对于诊断器理论来说，则应该关注监控器理论的最新发展，将监控器理论方面的最新成果运用进来，以取得进一步的成果。DEDS 故障诊断方法主要有以下几个研究方向：

（1）继续完善各个 DEDS 故障诊断方法的理论体系。

（2）发展和完善针对诸如分布式系统、混杂系统等大型复杂系统的 DEDS

故障诊断方法。

（3）结合其他定性方法（比如人工智能）和定量方法（比如解析冗余法），充分利用各种诊断方法的优点，为有效分析和控制故障提供更好的理论指导。

（4）应用方面的研究，比如智能公路系统、计算机/通信网络、自动化生产、集成电路制造、C^3I 系统等方面。

对于我们在文中着重论述的诊断器理论，则有许多有待进一步研究的东西，列举如下：

1、诊断器理论本身有许多方面值得进一步研究。这里举出几个：

（1）应该在什么层次上建立系统的离散事件模型（比如，建立一个模型应该包含多少状态）。

（2）如何在实现诊断的离散层次和连续值的传感器测量之间建立接口（即怎样将传感器输出离散化）。

（3）如何在离散事件层次处理未建模的动态情况（由于传感器噪音而产生的没有被建模的事件）。

（4）间歇故障的处理。

（5）针对分布式系统的故障诊断问题。

（6）当系统由大量相互作用组件构成时，怎样降低计算复杂性。（比如采用适当方法将系统分成易处理相互间低耦合度的子系统）

（7）如何将活跃诊断方法应用于更多类型的系统。

（8）运用诊断器理论处理混杂系统。

2、探测问题。探测的概念与我们前面讨论的诊断不同，诊断是靠被动的观察，而探测则是设计一个测试序列，主动发现故障。探测在微电子电路方面越来越重要，比如芯片的自检。探测问题原来经常使用基于模型的人工智能推理方法来解决的，不过现在也开始采用 DES 方法来产生探测序列。

3、本文中所研究的内容大都采用非时序模型。不过在许多应用中，时间是故障诊断中需要考虑的一个重要因素。在这种情况下，应该采用 DES 的赋时语言模型。不过已经有人将诊断器方法扩展到时间模型上，其中所用到的时间模型则是基于赋时自动机模型的。现在已经有研究表明，对于特定的时序 DES 系统，只要引入新的转移来建模时间片，就可以构建出等价的非时序模型，这样就能更好

的运用诊断器方法了。当然，有更多的研究需要进行，比如如何将诊断器理论新的研究成果运用到时序系统中。

4、结合其他 DEDS 故障诊断方法或者其他故障诊断方法。目前比较经常应用的 DEDS 故障诊断方法有基于状态的自动机方法，Petri 网方法，时序 DEDS 方法等。它们也具有各自的优点。各种方法相结合能很好处理类似混杂系统的一类复杂系统。

5、诊断器方法的应用问题，虽然已经运用到很多方面，但大都使用了集中式诊断的方法，很少见到采用分散诊断和针对分布式系统方面的应用。

6、相关的仿真软件。

所有这一切都有待于研究的进一步开展和深入。

参 考 文 献

【1】 P. J. Ramadge and W. M. Wonham, Supervisory control of a class of discrete event processes, SIAM J. Contr. Optim. , 1987, 25 (1). 206-230.

【2】 W. M. Wonham and P. J. Ramadge, On the supremal controllable sublanguage of a given language, SIAM J. Contr. Optim. , 1987, 25(3), 637-659.

【3】 Randy Cieslak, C. Desclaux, Ayman S. Fawaz, and Pravin Varaiya, Supervisory control of discrete-event process with partial observations, IEEE Trans. Automatic Control, 1988, 33(3), 249-259.

【4】 F. Lin, W. M. Wonham, On observability of discrete event systems, Infor. Sci., 1988, 44, 173-198.

【5】 P. J. Ramadge and W. M. Wonham, The control of discrete event systems, IEEE Proc., 1989, 77(4), 81-97.

【6】 S. Lafortune and E. Chen, The infimal closed controllable superlanguage and its application in supervisory control. IEEE Trans. Automat. Contr., 1990, 35(4), 398-405.

【7】 R.D. Brandt, V. Garg, R. Kumar, F. Lin, S.I. Marcus, and W. M. Wonham, Formulas for calculating supremal controllable and normal sublanguages, System & Control Letters, 1990, 15, 111-117.

【8】 K. Rudie and W. M. Wonham, The infimal prefix-closed and observable superlanguage of a given language, System & Control Letters, 1990, 15, 361-371.

【9】 R. Kumar, V. Garg, and S. I. Marcus, On controllability and normality of discrete event dynamical systems, System & Control Letters, 1991, 17, 157-168.

【10】 K. Inan, An algebraic approach to supervisory control, Math. Contr. Signa. Syst., 1992, 5, 151-164.

【11】 J. H. Fa, Y. P. Zheng, Formulas for a class of controllable and observable sublanguage larger than the supremal controllable and naormal sublanguage, Syst. Contr. Lett., 1993, 20, 11-18.

【12】 颜文俊，离散事件动态系统的监控理论及其应用研究，1994，浙江大学博士学位论文。

【13】 郑应平，离散事件系统理论研究和应用进展，控制与决策，1996，11（2），234-241。

【14】 R. Kumar and M. A. Shayman, Formula relating controllability, observability, and co-observability, Automatica, 1998, 34(2), 211-215.

【15】 F. Charbonnier, H. Alla, and R. David, The supervised control of discrete-event dynamics systems, IEEE Trans. Contr. Syst. Tech., 1999, 7(2), 175-187.

【16】 郑大钟，赵千川，离散事件动态系统，北京：清华大学出版社，2000。

【17】 F. Lin, W. M. Wonham, Decentralized control and coordination of discrete-event systems with partial observation, IEEE Trans. Automat. Contr., 1990, 35(12), 1330-1337.

【18】 Karen Rudie and W. M. Wonham, Think Globally, Act Locally: Decentralized Supervisory Control, IEEE Trans. Automat. Contr., 1992, 37(11), 1692-1707.

【19】 杨小军，郑应平，部分同步离散事件系统的分散监控，自动化学报，1992，18（6），728-732。

【20】 F. Lin and H. Mortazavian, A Normality Theorem for Decentralized Control of Discrete-Event Systems, IEEE Trans. Automat. Contr., 1994, 39(5), 1089-1093.

【21】 P. Kozak and W. M. Wonham, Fully decentralized solutions of supervisory control problems, Trans. Automat. Contr., 1995, 40(12), 2094-2097.

【22】 S. Takai, T. Ushio, and S. Kodama, Decentralized supervisory control using partial event and state observations, In Proc. 34th Conf. Decision and Cont., New Orleans, LA, December 1995, 723-724.

【23】 S. L. Ricker and K. Rudie, Know means no: incorporating knowledge into decentralized discrete-event control, In Proc.Amer. Contr. Conf., Albuquerque, New Mexico, June 1997, 2348-2353.

【24】 S. H. Lee and K. C. Wong, Decentralized control of concurrent discrete-event systems with non-prefix closed local specifications, In Proc. 36th Conf. Decision and Cont., San Diego, California USA, December 1997, 2958-2963.

【25】 K. H. Cho and J. T. Lim, Mixed centralized/decentralized supervisory control of discrete event dynamic systems, Automatica, 1999, 35, 121-128.

【26】 S. Jiang and R. Kumar, Decentralized control of discrete event systems with specializations to local control and concurrent systems, IEEE Trans. Syst. Man. Cyer. Part B: Cybernetics, 2000, 30(5), 653-660.

【27】 G. Barrett and S. Lafortune, Decentralized supervisory control with communicating controllers, IEEE Trans. Automat. Contr., 2000, 45(9), 1620-1638.

【28】 S. L. Ricker and K. Rudie, Know means no: incorporating knowledge into discrete-event control systems, IEEE Trans. Automat. Contr., 2000, 45(9), 1656-1668.

【29】 S. Takai and T. Ushio, On-line decentralized supervisory control of discrete event systems, In Proc. 39th Conf. Decision and Cont., Sydney, Australia, December 2000, 7-8.

【30】 T. Yoo and S. Lafortune, New Results on Decentralized Supervisory Control of Discrete-Event Systems, IEEE Proc. Decision. Contr., Sydney, Australia, December 2000,1-6.

【31】 吴旋，颜文俊，用于分散监控的联合正态理论，控制理论与应用（已投），2001.8。

【32】 N. B. Hadj-Alouane, S. Lafortune, and F. Lin, Think globally, communicate, act locally: on-line parallel/distributed supervisory control, In Proc. 33rd Conf. Decision and Cont., Lake Buena Vista, FL, December 1994, 3661-3666.

【33】 K. C. Wong, J. G. Thistle, R. P. Malhame, and H. H. Hoang, Supervisory control of distributed systems: conflict resolution, In Proc. 37th Conf. Decision and Cont., Tampa, Florida USA. December 1998, 3275-3280.

【34】 K. Rudie. S. Lafortune, and F. Lin, Minimal communication in a distributed discrete-event control system, In Proc. Amer. Contr. Conf., San Diego, California, June 1999, 1965-1970.

【35】 K. C. Wong, J. G. Thistle, R. P. Malhame, and H. H. Hoang, Supervisory control of distributed systems: Conflict Resolution, J. DEDS, 2000, 10. 131-186.

【36】 S. L. Ricker and K. Rudie, Distributed knowledge for communication in decentralized discrete-event systems, In Proc. 39th Conf. Decision and Cont., Sydney, Australia, December 2000, 9-15.

【37】 E. Fabre, A. Benveniste, C. Jard, L. Ricker, and M. Smith. Distributed state reconstruction for discrete event systems, In Proc. 39th Conf. Decision and Cont., Sydney, Australia, December 2000. 2252-2257.

【38】 K. C. Wong and W. M. Wonham, Modular control and coordination of discrete-event systems. J.DEDS. 1998, 8,247-297.

【39】 M. H. Queiroz and J. E. R. Cury. Modular control of composed systems, In Proc.Amer.

Contr. Conf., Chicago. Illinois, June 2000, 4051-4055.

【40】 R. Kumar, A discrete event systems approach for protocol conversion, J.DEDS, 1997, 7, 295-315.

【41】 郑应平，范中，离散事件动态系统理论在通信网络中的应用，系统工程理论与实践，1998, 9.

【42】 V. Minzu and D. C. Cernega, Supervisory control technique for assembly workstation, IEEE Proc. Symp. Assembly and Task Planning, Porto, Portugal, July 1999, 88-93.

【43】 L. H. Fink, Discrete events in power systems, Kluwer Academic Publishers, Boston, 1999, 319-330.

【44】 F. Lin, Diagnosibility of discrete-event systems and its applications, J.DEDS, 1994, 4 (2), 197-212.

【45】 P. Struss, Testing for discrimination of diagnosis, Proc. DX'94: International Workshop on Principles of Diagnosis, October 1994, 321-320.

【46】 L. Holloway and S. Chand, Time templates for discrete event fault monitoring in manufacturing systems, In Proc. Amer. Contr. Conf., 1994, 701-706.

【47】 S. Bavishi and E. Chong, Automated fault diagnosis using a discrete event systems framework, In Proc. 9th IEEE Int. Symp. Intelligent Contr., 1994, 213-218.

【48】 M. Sampath, A Discrete event systems approach to failure diagnosis, PhD thesis, Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI, 1995.

【49】 M. Sampth, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis, Diagnosibility of discrete event systems, IEEE Trans. Automat. Contr., 1995, 40(9), 1555-1575.

【50】 M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. C. Teneketzis, Failure diagnosis using discrete-event models, IEEE Trans. Contr. Systems. Tech., 1996, 4(2), 105-124.

【51】 T. Y. L. Chun, Diagnostic supervisory control: A discrete event systems approach, Master's thesis, Dept. of Elec. Eng., Univ. Of Toronto, 1996.

【52】 A. A. Nanavati, Designing Diagnosable Distributed Programs, PhD thesis, Department of Computer Science, Louisiana State University, May 1996.

【53】 M. Sampath, S. Lafortune, and D. Teneketzis, Active diagnosis of discrete event systems, IEEE Trans. Automatic Control, 1998, 43 (7), 908-929.

【54】 G. Provan and Y. L. Chen, Diagnosis of timed discrete event systems using temporal causal networks: Modeling and analysis. In Proc. Of the 1998 International Workshop on Discrete Event Systems (WODES'98), 1998, 152-154.

【55】 M. Larsson, Diagnosis and analysis of diagnosis properties using discrete event dynamic systems. In Proc. 37th IEEE Conf. On Decision and Control, December 1998,3775-3780.

【56】 S. H. Zad, R. H. Kwong and W. M. Wonham, Fault diagnosis in discrete-event systems: framework and model reduction, In Proc. 37th Conf. Decision and Cont., Tampa, Florida USA, December 1998, 3769-3774.

【57】 G. Provan and Y. L. Chen, Model-based diagnosis and control reconfiguration for discrete event systems: an integrated approach, In Proc. 38th IEEE Conf. On Decision and Control, 1999, 1762-1768.

【58】 G. Lamperti and M. Zanella, Diagnosis of discrete event systems integrating synchronous and asynchronous behavior, In Proc. 9th International Workshop on Principles of Diagnosis. DX'99, 1999,129-139.

【59】 S. H. Zad, Fault Diagnosis in Discrete-event and hybrid systems, PhD thesis, Dept. of Electrical and Computer Engineering, University of Toronto. September 1999.

【60】 高梅梅，吴智铭，模糊推理 Petri 网及其在故障诊断中的应用，自动化学报，2000，26（5），677－680。

【61】 R. Debouk, S. Lafortune, and D. Teneketzis, Coordinated decentralized protocols for failure diagnosis of discrete event systems, J.DEDS. 2000, 10 (1/2), 33-86.

【62】 R. Debouk, S. Lafortune, and D. Teneketzis, On the effect of communication delays in failure diagnosis of decentralized discrete event systems, In Proc. 39th Conf. Decision and Cont., Sydney, Australia, December 2000, 2245-2251.

【63】 S. Lafortune, D. Teneketzis, M. Sampath, R. Sengupta, and K. Sinnamohideen, Failure diagnosis of dynamic systems: an approach based on discrete event systems, In Proc.Amer. Contr. Conf., Arlington, VA, June 2001, 2058-2071.

【64】 颜文俊，吴旋，孙优贤，信息分散系统的活跃诊断方法，高校应用数学学报（已投），2002.1.

【65】 E. Fabre, A. Aghasaryan, A. Benveniste, R. Boubour, and C. Jard, Fault detection and diagnosis in distributed systems: an approach by partially stochastic petri nets, J. DEDS, 1998, 8, 203-231.

【66】 P.Baroni, G. Lamperti, P. Pogliano, and M. Zanella, Diagnosis of active systems, Artif. Intell., 1999, 110, 135-183.

【67】 P. Baroni, G. Lamperti, and M. Zanella, Diagnosis of a class of distributed discrete-event systems, IEEE Trans. System, man and cybernetics-Part A: systems and humans, 2000, 30(6), 731-752.

【68】 G. Lamperti and M. Zanella, Diagnosis of active systems by automata-based reasoning techniques, Applied Intelligence, 2000, 12, 217-237.

【69】 S. L. Ricker and E. Fabre, on the construction of modular observers and diagnosers for discrete-event systems, In Proc. 39th Conf. Decision and Cont., Sydney, Australia, December 2000, 2240-2244.

【70】 吴旋，颜文俊，一类分布式系统的模块化诊断方法，控制理论与应用（已投），2001.11。

【71】 S. H. Zad, R. H. Kwong , and W. M. Wonham, Fault diagnosis in timed discrete-event systems, In Proc. 38th Conf. Decision and Cont., Phoenix, Arizona USA, December 1999, 1756-1761.

【72】 J. Lunze, Diagnosis of quantized systems based on a timed discrete-event model, IEEE Trans. Syst. Man. Cyber. Part A: System and Humans, 2000, 30(3), 322-335.

【73】 N. Viswanadham and T. L. Johnson, Fault detection and diagnosis of automated manufacturing systems, In Proc. 27th IEEE Conf. On Decision and Control, Austin Texas, 1988, 2301-2306.

【74】 A. T. Bouloutas, Modeling fault management in communication networks, PhD thesis, Columbia University, 1990.

【75】 M. Sampath, Discrete event systems based diagnostics for a variable air volume terminal box application, Technical report, Advanced Development Team, Johnson Controls, Inc., September 1995.

【76】 B. T. Murray and J. P. Hayes, Testing ICs: getting to the core of the problem, IEE Computer, 1996, 29(11), 32-38.

【77】 Y. L. Chen and G. Provan, Modeling and diagnosis of timed discrete event systems – a

factory automation example, In Proc. 1997 American Control Conf., Albuquerque, NM, June 1997, 31-36.

【78】 D. N. Godbole and R. Sengupta, Tools for the design of fault management systems, In Proc. of the IEEE Intelligent Transportation Systems Conference, 1997, 159-164.

【79】 L. Roze, Supervision de reseaux de telecommunications: Une approche a base de modeles, PhD thesis, Universite de Rennes I, France, 1997.

【80】 R. Boubour, C. Jard, A. Aghasaryan, E. Fabre, and A. Benveniste, A Petri net approach to fault detection and diagnosis in distributed systems. Part I: application to telecommunication networks, motivations, and modeling, In Proc. 36th Conf. Decision and Cont., San Diego, California USA, December 1997, 720-725.

【81】 G. Westerman, R. Kumar, C. Stroud, and J. R. Heath, Discrete event system approach for delay fault analysis in digital circuits, In Proc. 1998 American Control Conf., 1998.

【82】 L. Roze and M. O. Cordier, Diagnosing discrete-event systems: An experiment in telecommunication networks, In Proc. of the 1998 International Workshop on Discrete Event Systems (WODES'98), 1998, 130-137.

【83】 R. Sengupta, Diagnosis and communication in distributed systems, In Proc. of the 1998 International Workshop on Discrete Event Systems (WODES'98), 1998, 144-151.

【84】 M. Sampath, Embedded print engine diagnostics: The dc265 project & beyond, Technical Report Xerox Internal Report: X9900094, Xerox Corporation, April 1999.

【85】 H. T. Simsek, R. Sengupta, S. Yovine, and F. Eskafi, Fault diagnosis for intra-platoon communication, In Proc. 38th IEEE Conf. On Decision and Control, December 1999.

【86】 J. Lunze, Diagnosis of quantised systems, In IFAC Safe Process 2000, 2000, 28-39.

【87】 Y. Pencole, Decentralized diagnosers approach: Application to telecommunication networks, Proc. DX'00: 11th International Workshop on Principles of Diagnosis, June 2000, 185-192.

【88】 M. Sampath, A. Godambe, E. Jackson, and E. Mallow, Combining qualitative and quantitative reasoning – a hybrid approach to failure diagnosis of industrial systems, In IFAC Safe Process 2000, June 2000, 494-501.

【89】 R. Sengutpa, Discrete-event diagnostics of automated vehicles and highways, In Proc. 2001 American Control Conf., June 2001.

【90】 M. Sampath, A hybrid approach to failure diagnosis of industrial systems, In Proc. 2001 American Control Conf. June 2001.

【91】 K. Sinnamohideen, Discrete-event diagnostics of heating, ventilation, and air-conditioning systems, In Proc. 2001 American Control Conf., June 2001.

【92】 S. J. Park and J. T. Lim, Fault-tolerant robust supervisor for discrete event systems with model uncertainty and its application to a workcell, IEEE Trans. Robot. Automat. , 1999, 15(2), 386-391.

【93】 L. Magnif, R. Scattolinif, and C. Rossif, A fault detection and isolation method for automotive engines, In Proc. Conf. Advanced Intelligent Mechatronics, Atlanta, USA, September 19-23 1999, 263-268.

【94】 D. N. Godbole, J. Lygeros, E. Singh, A. Deshpande, and A. E. Lindsey, Communication protocols for fault-tolerant automated highway system, IEEE Trans. Contr. Syst. Tech., 2000, 8(5), 787-800.

【95】 S. Takai and T. Ushio, Relaible decentralized supervisory control of discrete event systems, IEEE Trans. Syst. Man. Cybe. Part B: Cybernetics, 2000, 30(5), 661-667.

【96】 L. Magni, R. Scattolini, and C. Rossi, A fault detection and isolation method for complex industrial systems, IEEE Trans. Syst. Man. Cybe. PartA: Systems and Humans, 2000, 30(6), 860-865.

【97】 P. M. Frank, Analytical and qualitative model-based fault diagnosis – a survey and some new results, European Journal of Control, 1996, 2, 6-28.

【98】 J. Gertler, Fault Detection and diagnosis in engineering systems, Marcel Dekker, 1998.

【99】 A. D. Pouliezos and G. S. Stavrakakis, Real time fault monitoring of industrial processes, Kluwer Academic Publishers, 1994.

【100】 A. D. Pouliezos and G. S. Stavrakakis, Real time fault monitoring of industrial processes, Kluwer Academic Publishers, 1994.

【101】 R. De Vrise, An automated methodology for generating a fault tree, IEEE Ttrans. Reliability Engineering, 1990, 39(1), 76-86.

【102】 H. Milde and L. Holtz, Facing diagnosis reality-model based fault tree generation industrial applications, Proc. DX'00: 11th International Workshop on Principles of Diagnosis, June 2000, 147-154.

【103】 A. Darwiche and G. Provan, Exploiting system structure in model-based diagnosis of discrete event systems, In Proceedings of the Seventh International Workshop on the Principles of Diagnosis, DX-96, Val Morin, Canada, 1996.

【104】 W. Hamscher, L. Console, and J. Kleer, Rdadings in Model Based Diagnosis, Morgan Kaufmann, 1992.

【105】 B. Williams and P. Nayak, A model-based approach to reactive self-configuring systems, In Proceedings of the AAAI, 1996.

附录 A SCPO 问题

在系统部分能观的情况下，给出一个语言的范围，要求设计一个监控器，使得系统的闭环行为在这个范围内。这类监控器设计问题就称为 SCPO（Supervisory Control Problem with Observation）问题。假设给出一个语言范围：$\bar{A} \subseteq L(\Psi/G) \subseteq \bar{E}$。下面的定理解决了这个问题：

定理 A.1：存在一个完备监控器 Ψ 满足 $A \subseteq L_c(\Psi/G) \subseteq E$，且 $\bar{A} \subseteq L(\Psi/G) \subseteq \bar{E}$，当且仅当 $\inf \underline{CO}(\bar{A}) \subseteq \bar{E}$ 时。

在定理 A.1 中，$\underline{CO}(L) \triangleq \{K : K \supseteq L \wedge \bar{K}\Sigma_{uc} \cap L(G) \subseteq \bar{K} \wedge K \text{ 能观}\}$。因此，只要满足 $\inf \underline{CO}(\bar{A}) \subseteq \bar{E}$，就可以构造监控器使得 $L(\Psi/G) = \inf \underline{CO}(\bar{A})$。

命题 A.1：$\inf \underline{CO}(L) = L(G) - (\Sigma^*\Sigma_c - M)\Sigma^*$，其中

$$M \triangleq \bigcup_{\sigma \in \Sigma_c} (P^{-1}(P(\bar{L}\sigma \cap \bar{L})) \cap \Sigma^*\sigma)。$$

由命题 A.1 可以得出下面的推论：

推论 A.1：$\inf \underline{CO}(L) = (L(G) - \Sigma^*\Sigma_c\Sigma^*) \cup (L(G) \cap P^{-1}(P(\bar{L}\Sigma_c \cap \bar{L}))\Sigma^*)$

下面举一个 SCPO 问题的例子来说明。设有一个系统如图 A.1 所示，$\Sigma = \{\alpha, \beta, \gamma\}$，$\Sigma_c = \{\alpha, \gamma\}$，并假定要求达到的监控行为是闭的，即系统的所以状态都是可标记的。

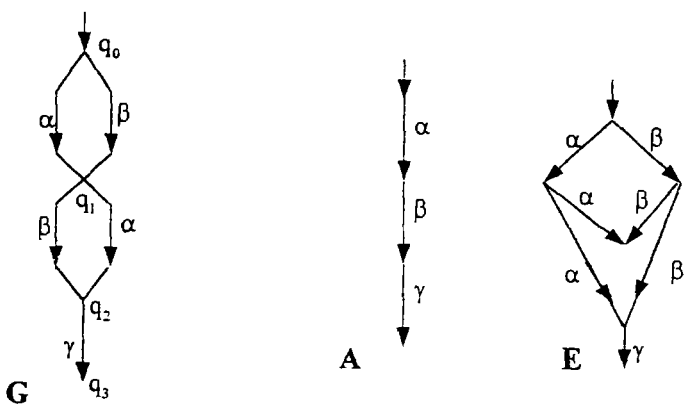
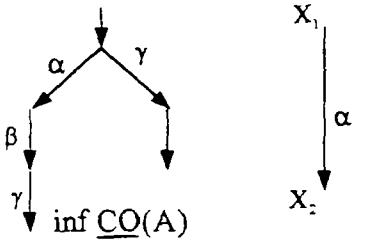


图 A.1 简单系统及其合法行为和最大容许行为

由图 A.1 可知，$L(G) = \overline{(\alpha+\beta)(\alpha+\beta)\gamma}$，$A = \overline{\alpha\beta\gamma}$，$E = \overline{(\alpha\beta+\beta\alpha)\gamma} + \overline{\alpha^2} + \overline{\beta^2}$。

而 A 不能控，不能观。根据推论 A.1 可以求出：$\inf \underline{CO}(A) = \overline{\alpha\beta\gamma} + \overline{\beta^2}$。可以得出：$A \subseteq \inf \underline{CO}(A) \subseteq E$，根据定理 A.1，存在监控器。

由于$\phi \neq A \neq \inf \underline{CO}(A) \neq E \neq L(G)$，所以该解是非平凡的，算法有实际意义。这样就可以构造监控器Ψ，使得$L_c(\Psi/G) = L(\Psi/G) = \inf \underline{CO}(A)$。想要的优化语言和监控器如图 A.2 所示，其中监控器$\Psi = (S, \varphi)$。



S

事件 / 状态	α	γ
X_1	1	0
X_2	0	1

φ

图 A.2 优化语言和监控器

如果想了解更多关于的 SCPO 问题的内容，请参考文献【5，12，16】。

附录B 监控器综合中的子语言算法介绍

为了有效设计监控器，需要对给定语言进行优化和逼近，以符合监控器存在的条件。因此经常要求解一定条件下的子语言。所以，在监控理论中，子语言算法有着比较重要的地位。下面我们将介绍一些主要的子语言及一些算法。

B.1 给定语言的能控子语言的上确界

在2.2.3的讨论中可知，在设计监控器的时候，如果期望语言K不能满足能控性，可以用K的最大能控子集来逼近。所以定义：

$$\sup C(L) \triangleq \bigcup \{K : K \subseteq L \wedge \bar{K}\Sigma_{uc} \cap M \subseteq \bar{K}\}$$

记为$L^{\uparrow}$，用它逼近L。本文中，M=L(G)。根据$\underline{C}(L)$的定义可知，

$$\sup \underline{C}(L) \triangleq \bigcup \{K : K \subseteq L \wedge K = \bar{K} \wedge \bar{K}\Sigma_{uc} \cap M \subseteq \bar{K}\}$$

$\uparrow$算子具有下面的性质：

定理 B.1[2]：（1）$L^{\uparrow} = \overline{L^{\uparrow}} \cap L$；（2）若$L = \bar{L} \cap L_m(G)$，则$L^{\uparrow} = \overline{L^{\uparrow}} \cap L_m$；

（3）$(L^{\uparrow})^{\uparrow} = L^{\uparrow}$；$(L_1 \cup L_1)^{\uparrow} \supseteq L_1^{\uparrow} \cup L_2^{\uparrow}$，$(L_1 \cap L_1)^{\uparrow} \subseteq L_1^{\uparrow} \cap L_2^{\uparrow}$。

下面从3个方面论述supC(L)、$\sup \underline{C}(L)$的求法及一些性质。

B.1.1 当给定语言不闭时

可以引入一个算子$\Omega^{\uparrow}(K) \triangleq \{t : t \in L \text{且} \bar{t}\Sigma_{uc} \cap M \subseteq \bar{K}\}$。该算子有如下性质：

定理 B.2[12]：（1）令S=supC(L)，则$S = \Omega^{\uparrow}(S)$，且对每一个满足$K = \Omega^{\uparrow}(K)$的语言，都有$S \supseteq K$；（2）若$P \subseteq Q \subseteq L$，则$\Omega^{\uparrow}(P) \subseteq \Omega^{\uparrow}(Q)$。

由定理B.1和定理B.2，可以定义如下序列：

$K_0 = L$，$K_{j+1} = \Omega^{\uparrow}(K_j)$，j=0,1,…，（B-1）

则有如下结论[2]：

定理 B.3：（1）集合极限$K \triangleq \lim_{j \to \infty} K_j$存在且$L^{\uparrow} \subseteq K$；

（2）如果L、M是正规的，则经过有限步递推后有$L^{\uparrow} = K$，且$L^{\uparrow}$也是

正规的。

由定理 B.2 可知，当 L、M 是正规的时候，可以采用这个序列得到$L^{\uparrow}$，当然，这个条件是充分条件，而非必要的。

B.1.2 当给定语言闭时

当给定语言闭时，B.1.1 中方法仍然可以使用，另外，根据能控性定义，有下面推论：

推论 B.1[12]：令$K=\bar{K}\subseteq L(G)$，$s\in K^{\uparrow}$且$s\sigma\in L(G)$，则$s\sigma\notin K^{\uparrow}\Leftrightarrow s\sigma l\in L(G)-K$，其中$l\in\Sigma_{uc}^{*}$。

该推论表明，如果$s\in K^{\uparrow}$，为了阻止σ的发生，即$s\sigma\notin K^{\uparrow}$，则在 L(G)-K 中存在一个不可控后缀字符串。因此，为了求解 K 中的最大能控子语言，就必须去掉这类字符串。令$D_{uc}(s)$为 s 中最后事件是可控事件的最长前缀；则$D_{uc}(L(G)-K)$就包含了所有可能导致$s\sigma\notin K^{\uparrow}$的字符串，所以 K 的最大能控子语言和最大能控闭子语言如下[7]：

$K^{\uparrow}=K-D_{uc}(L(G)-K)$（B-2），$\sup\underline{C}(K)=K-D_{uc}(L(G)-K)\Sigma^{*}$（B-3）。

可以证明，当 K、L(G)正规时，$K^{\uparrow}$和$\sup\underline{C}(K)$可以有效计算。

由上面的式子可得，当给定语言 K 不闭时，$\sup\underline{C}(K)=\underline{K}-D_{uc}(L(G)-\underline{K})\Sigma^{*}$（B-4）。其中：$\underline{K}\triangleq\{s\in K:\{\bar{s}\}\in K\}$（B-5），即 K 的最大闭子语言。

（3）最大能控子语言的估计

当给定子语言不闭时，除了 I 中的方法外，还可以利用闭能控子语言的特性对$K^{\uparrow}$进行估计。有下面推论[12]：

推论 B.2：$\underline{K}\subseteq K\subseteq\bar{K}$，$(\underline{K})^{\uparrow}\subseteq K^{\uparrow}\subseteq(\bar{K})^{\uparrow}\cap K$；

推论 2.3 给出了$K^{\uparrow}$的上界和下界，所以可以很方便的估计出$K^{\uparrow}$。特别是当$(\underline{K})^{\uparrow}=(\bar{K})^{\uparrow}$时，可以直接得出$K^{\uparrow}$。

B.2 给定语言的能控子语言的下确界

与能控子语言的上确界相对应，可以定义如下语言类：

$$C_p(L) \triangleq \{K \subseteq \Sigma^* : (K \supseteq L) \wedge \bar{K}\Sigma_{uc} \cap M \subseteq \bar{K} \wedge K = \bar{K}\}$$

根据能控的性质，$C_p(L)$ 的下确界 $\inf C_p(L)$ 也是存在的，定义如下：

$$\inf C_p(L) \triangleq \bigcap \{K \subseteq \Sigma^* : (\bar{L} \subseteq K \subseteq M) \wedge \bar{K}\Sigma_{uc} \cap M \subseteq \bar{K} \wedge K = \bar{K}\}$$

记为 $L^{\downarrow}$。$\downarrow$ 算子具有如下性质[12]：

定理 B.4：

（1）$\overline{L^{\downarrow}} = \bar{L}^{\downarrow}$，$(L^{\downarrow})^{\downarrow} = L^{\downarrow}$；

（2）$(L_1 \cup L_2)^{\downarrow} = L_1^{\downarrow} \cup L_2^{\downarrow}$；

（3）$(L_1 \cap L_2)^{\downarrow} \subseteq L_1^{\downarrow} \cap L_2^{\downarrow}$，当 L_1、L_2 闭或者非排斥的时候，等号成立。

综合定理 B.1 和 B.4，两个算子有如下性质：

定理 B.5：

（1）$(L^{\uparrow})^{\downarrow} \supseteq L^{\uparrow}$；$(L^{\downarrow})^{\uparrow} = L^{\downarrow}$；$(L^{\uparrow})^{\downarrow} \subseteq (L^{\downarrow})^{\uparrow}$；

（2）$(L_1^{\uparrow} \cap L_2^{\downarrow})^{\uparrow} = L_1^{\uparrow} \cup L_2^{\downarrow}$；$(L_1^{\uparrow} \cap L_2^{\downarrow})^{\uparrow} = L_1^{\uparrow} \cap L_2^{\downarrow}$，如果 L_1 闭；

（3）如果 L_1 闭，$(L_1^{\uparrow} \cup L_2^{\downarrow})^{\downarrow} = L_1^{\uparrow} \cup L_2^{\downarrow}$，$(L_1^{\uparrow} \cap L_2^{\downarrow})^{\downarrow} \subseteq L_1^{\uparrow} \cap L_2^{\downarrow}$；

（4）$(L_1^{\uparrow} \cup L_2^{\downarrow})^{\downarrow} \subseteq L_1^{\downarrow} \cup L_2^{\downarrow}$，$(L_1^{\uparrow} \cap L_2^{\downarrow})^{\downarrow} \subseteq L_1^{\downarrow} \cap L_2^{\downarrow}$。

为了求 $\inf C_p(L)$，令 K、M 闭，定义算子 $\Omega^{\downarrow}$ 如下[6]：

$$\Omega^{\downarrow}(K) \triangleq \bigcap \{T : T \subseteq \Sigma^* \wedge T = \bar{T} \wedge T \supseteq L \wedge K\Sigma_{uc} \cap M \subseteq T\}$$

该算子具有下面的性质：

命题 B.1[12]：（1）$L^{\downarrow} = \Omega^{\downarrow}(L^{\downarrow})$，且对每一个满足 $K=\Omega^{\downarrow}(K)$ 的语言，都有 $L^{\downarrow} \supseteq K$；

（2）若 $P \subseteq Q \subseteq \bar{L}$，则 $\Omega^{\downarrow}(P) \subseteq \Omega^{\downarrow}(Q)$。

很容易得到下面的引理：

引理 B.1[6]：$\Omega^{\downarrow}(K) = \bar{L} \cup \overline{K\Sigma_{uc} \cap M}$。

当 $\Sigma_o = \Sigma$ 时，有 $L^{\downarrow} = M \cap \bar{L}\Sigma_{uc}^*$（2-6）。

这个算子也可以用下面递推公式获得：

$K_0 = \bar{L}$， $K_{j+1} = \Omega^{\downarrow}(K_j) = \bar{L} \cup \overline{K_j \Sigma_{uc} \cap M} = K_j \cup (K_j \Sigma_{uc} \cap M)$， j=0,1,2…（B-7）

而 $K_\infty = \bigcup_j K_j \supseteq \bar{L}$， $K_\alpha \subseteq M$，所以 K_α 存在。下面定理给出了 $L^{\downarrow}$ 的求法：

定理 B.6[12]：递推算法（B-7）有相同的结果，且 $K_\infty = L^{\downarrow}$。

不过在 L、M 为正规语言的情况下，算法 2-7 可能不会在有限步内收敛。这时如果‖L‖和‖M‖不太大，可以用 2-6 式直接求，否则可以用下面的自动机构造方法求解：

算法 B.1：

（1） 构造 $H = (Q, \Sigma, \delta, q_0)$ 和 $G = (X, \Sigma, \xi, x_0)$，使得 L(H)=L，L(G)=M；

（2） 对 $\forall q \in Q$ 和 $\forall \sigma \in \Sigma_{uc}$，如果 $\sigma \notin \{\sigma \in \Sigma : \delta(\sigma, q)!\}$，则在 δ 中增加转移 (q, δ, q_d)，由此得到 H_1；

（3） 求得有限自动机的交 $H_1 \cap G$，则 $L(H_1 \cap G) = L^{\downarrow}$。

B.3 完全能观时的其他语言类

在完全能观的情况下，除了能控子语言 C(L)和 $\underline{C}(L)$ 以外，还有一些很重要的语言类，分别定义如下：

$F(L) \triangleq \{K \subseteq L : K = \bar{K} \cap L_m(G)\}$；

$NC(L) \triangleq \{K \supseteq L : \bar{K}\Sigma_{uc} \cap L(G) \subseteq \bar{K}\}$：

$FC(L) \triangleq \{K \subseteq L : K = \bar{K} \cap L_m(G) \wedge \bar{K}\Sigma_{uc} \cap L(G) \subseteq \bar{K}\}$：

supF(L)有三种方法可以求解，直接搜索法和自动机生成法可见【12】，下面是它的闭形解：

定理 B.7[12]：$\sup F(L) = L - (\bar{L} \cap L_m(G) - L)\Sigma^*$。

根据 supF(L)的闭形解，我们可以得出 supFC(L)的闭形解：

引理 B.2[12]：$\sup FC(L) = \sup C(\sup F(L)) = [L - (\bar{L} \cap L_m(G) - L)\Sigma^*]^{\uparrow}$。

当然，supFC(L)也可以采用递推公式求解：

$L_1 = L^{\uparrow}$， $L_{i+1/2} = L_i - (\bar{L}_i \cap L_m(G) - L_i)\Sigma^*$， $L_{i-1} = (L_{i+1/2})^{\uparrow}$， i=0,1,2,…（B-8）

定理 B.8[2]：（1）$L_\infty = \lim_{n\to\infty} L_n$；（2）$\sup FC(L) \subseteq L_\infty$；（3）当 L、$L_m(G)$ 正规时，$\sup FC(L) = L_\infty$ 并且它也是正规的。

至于 NC(L)，由于它在交运算下不封闭，所以下确界一般不存在。如果找到一组序列$\{K_i\}$，$\bar{K}_i\Sigma_{uc} \cap L(G) \subseteq \bar{K}_i$，且$L \subseteq K_1 \subseteq K_1 \subseteq K_1 \subseteq \cdots \subseteq L_m(G)$。根据 Zorn 引理，在 NC(L)中至少存在一最大元 maxNC(L)。

B.4 部分能观时的语言类

根据 2.2.3 中的论述，如果Σ中有些事件不能被监控器观测到，监控器存在的充要条件是期望行为能控、能观，因而此时监控器设计的主要问题是求得给定语言的能控、能观子语言。涉及到能观性的语言类，除了前面提到的$\underline{CO}(L)$外，还有下面的这些：

$\underline{O}(L) \triangleq \{K : K \subseteq L, K = \bar{K}$, 且 K 能观$\}$；

$\underline{O}'(L) \triangleq \{K : K \supseteq L, K = \bar{K}$, 且 K 能观$\}$；

$\underline{N}(L) \triangleq \{K : K \subseteq L, \bar{K}$ 是正态语言$\}$；

$\underline{CN}(L) \triangleq \{K : K \subseteq L, K = \bar{K}$ 且 K 是正态语言$\}$；

$RCN(L) \triangleq \{K : K \subseteq L, K = \bar{K} \cap L_m(G), \bar{K}$ 是正态语言$\}$；

$\underline{RCN}(L) \triangleq \{K : K \subseteq L, K = \bar{K}$ 且 K 是能控正态语言$\}$；

$Z(L) \triangleq \{K : K \subseteq L, K = \bar{K} \cap L_m(G), \bar{K}\Sigma_{uc} \cap L(G) \subseteq \bar{K}, \bar{K}$ 是正态语言$\}$；

$Z'(L) \triangleq \{K : K \subseteq L, K = \bar{K} \cap L_m(G), \bar{K}\Sigma_{uc} \cap L(G) \subseteq \bar{K}$ 且 K 能观$\}$；

$D(L) \triangleq \{K : K \subseteq L, \bar{K}\Sigma_{uc} \cap L(G) \subseteq \bar{K}, \bar{K} = L(G) \cap P^{-1}P(\bar{K})\}$ 且 K 能观$\}$；

$\underline{CO}'(L) \triangleq \{K : K \subseteq L, K = \bar{K}, \bar{K}\Sigma_{uc} \cap L(G) \subseteq \bar{K}$ 且 K 能观$\}$；

结合能控、能观、正态及$L_m(G)$的有关性质，可以得出下面结论：

定理 **B.9**：在上述语言类中，$\inf \underline{O}'(L)$、$\sup \underline{N}(L)$、$\sup \underline{CN}(L)$、supRCN(L)、$\sup \underline{RCN}(L)$、supZ(L)、$\inf \underline{CO}(L)$是存在的，而 $\sup \underline{O}(L)$、$\sup \underline{Z}'(L)$、$\sup \underline{CO}'(L)$一般是不存在的。

$\sup \underline{Z}'(L)$、$\sup \underline{CO}'(L)$不存在的原因是能观语言的并一般不再是能观语言。而在部分能观的情况下，监控器存在的充要条件是期望语言能控、能观。为了求得最小约束的优化解，需要最大限度逼近期望行为。所以，有的文献提出了用 $\underline{CO}'(L)$中最大元来代替上确界，但其求法比较烦琐，且不一定收敛。很多文献通常都用正态性来代替能观性，即采用 supRCN(L)代替 $\sup \underline{CO}'(L)$，不过正态性要求比能观严格。针对这种情况，文献【11】提出了一种介于二者之间的能控、能观子语言算式：

$$\Omega(L) = L - \{\bigcup_{\sigma \in \Sigma} L \cap L\sigma \cap P^{-1}P[(L(G) - L) \cap L\sigma]\}\Sigma^*$$

该算式保证任何闭语言 L，$\Omega(L)$ 是闭、能观的。且对于任意能控语言 L，$\Omega(L)$ 闭且能控。所以我们可以用 $\Omega(\sup C(L))$ 去逼近 $\underline{CO}'(L)$中的最大元，可以证明：

$$\sup \underline{RCN}(L) \subseteq \Omega(\sup C(L)) \subseteq \max(\underline{CO}'(L))。$$

至于 $\sup \underline{CN}(L)$、$\sup \underline{RCN}(L)$，有以下的闭解形式[7]：

$$\sup \underline{CN}(L) = L - P^{-1}P(L(G) - L)\Sigma^*；（B\text{-}9）$$

$$\sup \underline{RCN}(L) = L(G) \cap P^{-1}(P \sup \underline{CN}(L) - D_{wc}(PL(G) - P \sup \underline{CN}(L))\Sigma_o^*)；$$

（B-10）

$\inf \underline{O}'(L)$则有如下闭形解：$\inf \underline{O}'(L) = L(G) - (\Sigma^- - \tilde{P}^{-1}(\tilde{P}(L)))\Sigma^*$（B-11），其中 $\Sigma^- = \Sigma^* - \{\varepsilon\}$，$\tilde{P}^{-1}(\tilde{P}(L)) = \bigcup_{\sigma \in \Sigma}[P^{-1}(P(\bar{L}\sigma \cap \bar{L})) \cap \Sigma^*\sigma] \cup \{\varepsilon\}$。

对于 supRCN(L)、supZ(L)和 supD(L)，有如下定理：

定理 **B.10**[12]：若 L 是 $L_m(G)$ 闭，supD(L)= supZ(L)，supRCN(L)= $\sup \underline{N}(L)$。

由该定理可得：supRCN(L)=$\sup \underline{N}(\text{supF}(L))$，supZ(L)=supD(supF(L))。

附录 C 模块化监控方法

模块化设计方法是解决分散监控问题的基础。下面对这种方法进行叙述：

考虑一个离散事件系统 $G=(Q,\Sigma,\delta,q_0,Q_m)$，$\Sigma=\Sigma_{uc}\cup\Sigma_c$，系统的期望行为 $L\subseteq L(G)$ 是闭的，且 L 可以表示成：$L=\bigcap_{i=1}^{n}L_i$。要求设计监控器 S，使 $L(S/G)\subseteq L$。集中式监控方法都是先求出 L 的识别器 $R=(X,\Sigma,\xi,x_0,X_m)$，使得 $L_m(R)=L$，然后用【1】中的思想，构造 $\Psi=(S,\varphi)$ 满足 $L(S/G)\subseteq L$。但实际系统的期望行为（或约束）往往由许多规则产生的，即 L_i 已知，分别设计 Ψ_i，使 $L(\Psi_i/G)\subseteq L_i$，则有 $L(\Psi_i/G)\subseteq\bigcap_{i=1}^{n}L_i\subseteq L$。其中的监控器可以用如下方法构造：

令 $\Psi_i=(S_i,\varphi_i)$，S_i 为一自动机且有：

$S_i=(X_i,\Sigma,\xi_i,x_{0i},X_{mi})$，$S=\wedge_{i=1}^{n}S_i=(X,\Sigma,\xi,x_0,X_m)$，

$X=X_1\times X_2\times\cdots\times X_n$，$X_m=X_{m1}\times X_{m2}\times\cdots\times X_{mn}$，$x_0=x_{01}\times x_{02}\times\cdots\times x_{0n}$，

状态迁移函数：$\xi_1\times\xi_2\times\cdots\times\xi_n(\sigma,x_1,x_2,\cdots,x_n)$ =

$$\begin{cases}(\xi_1(\sigma,x_1),\xi_2(\sigma,x_2),\cdots,\xi_n(\sigma,x_n))，\text{如果}\xi_i(\sigma,x_i)!,1\le i\le n\\ \text{无定义，否则}\end{cases}$$

而联合监控器 $\Psi=(S,\varphi)$，$S=(X,\Sigma,\xi,x_0,X_m)$，$\varphi(x_1,x_2,\cdots,x_n)=\varphi(x_1)\bullet\varphi(x_2)\cdots\varphi(x_n)$。

下面的定理给出了非阻塞监控器存在的条件和构造方法[12]：

定理 C.1：若 $\overline{L_i^{\uparrow}\cap L_j^{\uparrow}}=\overline{L_i^{\uparrow}}\cap\overline{L_j^{\uparrow}}$，$L_i=\overline{L_i}\cap L_m(G)$，则存在监控器 S，使

$L_c(S/G)=\bigcap_{i=1}^{n}L_i^{\uparrow}$，$L(S/G)=\overline{L_c(S/G)}$。

通过采用模块化设计方法，计算量就大大减小。比如对于 n=1，L_1、L_2 闭的情况，假设 L_1、L_2 中元素各为 ℓ 个，Σ_{uc} 中有 m 个元素，则 $L_i^{\uparrow}$ 的复杂度为

$O(\ell^3 m^3)$，检验 $L_1^{\uparrow}$ 与 $L_2^{\uparrow}$ 的非排斥性的计算复杂度为 $O(\ell^4)$，因此 $L_1^{\uparrow} \cap L_2^{\uparrow}$ 的监控器集成的计算复杂度为 $O(\ell^3 m^3 + \ell^4)$，而直接计算 $(L_1 \cap L_2)^{\uparrow}$ 的复杂度为 $O(\ell^6 m^3)$。不过这种设计方法要求的条件比较严格，从而增加了实际应用中的困难。更详细的有关模块化方法的内容可参考文献【38】、【39】。

附录 D 作者攻读硕士研究生期间完成的论文

【1】 吴旋，颜文俊，轿车门锁自动装配系统中的钥匙代码自动识别系统，电气传动，Vol. 30, 2000.8。

【2】 吴旋，颜文俊，用于分散监控的联合正态理论，控制理论与应用（已投），2001.8。

【3】 吴旋，颜文俊，一类分布式系统的模块化诊断方法，控制理论与应用（已投），2001.11。

【4】 颜文俊，吴旋，孙优贤，信息分散系统的活跃诊断方法，高校应用数学学报（已投），2002.1。