离散事件系统的同步诊断算法

王晓宇[1,3] 欧阳丹彤[1,3] 赵相福[1,3] 常晓环[2,3]
(吉林大学计算机科学与技术学院 长春 130012)[1] (吉林大学软件学院 长春 130012)[2]
(吉林大学符号计算与知识工程教育部重点实验室 长春 130012)[3]

摘 要 针对传统系统建模方法需要假设模型完备的缺点,提出一种通过同步各部件模型的方法来解决不完备建模所导致的不完全诊断。对于离散事件系统的动态诊断进行优化,利用分布式的思想与Petri网的性质,使得各部件可以独立、并行地进行诊断,提高了诊断的速度。同时对提出的同步方法进行了可行性分析和简单实现,得到了较好的结果。

关键词 离散事件系统,动态诊断,Petri网,同步
中图法分类号 TP306 **文献标识码** A

Algorithm of Dynamic Event System's Synchronization Diagnosis

WANG Xiao-yu[1,3] OUYANG Dan-tong[1,3] ZHAO Xiang-fu[1,3] CHANG Xiao-huan[2,3]
(College of Computer Science and Technology, Jilin University, Changchun 130012, China)[1]
(College of Software, Jilin University, Changchun 130012, China)[2]
(Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University, Changchun 130012, China)[3]

Abstract Traditional system modeling requires the assumption that the model must be a complete one, according to this disadvantage, this paper proposed a method by synchronization components' model. The method synchronizes incomplete models to a larger one, to solve incomplete diagnosis caused by incomplete models. Optimizing dynamic diagnosis for discrete event systems, and with the idea of distributed system and the nature of Petri Nets, components can make diagnosis in an independent and parallel way, thus improve the rate of obtaining a diagnosis.

Keywords Discrete event system, Dynamic diagnosis, Petri net, Synchronization

基于模型诊断的动态方法是为了克服传统故障诊断方法的缺点而提出的一项新推理技术,其在静态诊断的基础上进一步扩展到系统状态的运行中[1,2]。离散事件系统作为诊断建模的一种思想方法,已广泛应用于各类系统上[3,4]。

现有的模型通常建立在完备性[2,5]的假设基础上,然而实际情况中假设并非总是成立。即使能够对单个部件建立一个完备模型,也不能将部件系统地组织在一起。如果系统的故障是由多个部件共同引起的,那么即使部件建模完备,它给出的诊断路径也是不完全的,这便产生了诊断的缺失。若是对于整个系统建模,庞大的状态和事件个数会使得诊断速度缓慢,甚至由于组合爆炸而不能得出诊断。

针对以上的问题,本文提出了一种分布式的诊断方法,这种方法的优点是:不需要假设建模完备;根据Petri网的工作流特性[6-8],利用分布式方法,建立单独的部件模型,每个部件独立进行诊断,只有诊断不能由独立部件给出时才对相关部件进行同步,得到同步机,从而给出完全的诊断信息。

1 相关定义

1.1 Petri网基本定义

定义 1(Petri网) 满足如下条件的三元组 $N=(S,T;F)$ 称作一个Petri网[9]:

(1) $S\cap T=\emptyset$;

(2) $F\subseteq(S\times T)\cup(T\times S)$;

(3) $S\cup T\neq\emptyset$;

(4) $\mathrm{dom}(F)\cup\mathrm{cod}(F)=S\cup T$。

其中,S 是状态集合,在图中表示成圆形节点,称作库所;T 是事件集合,在图中表示成矩形节点,称作变迁。$\mathrm{dom}(F)=\{x|\exists y:(x,y)\in F\}$,$\mathrm{cod}(F)=\{x|\exists y:(y,x)\in F\}$。

图1是一个Petri网的简单例子。

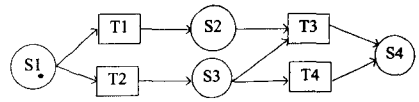

图1 Petri网示例

到稿日期:2009-03-31 返修日期:2009-06-04 本文受国家自然科学基金重大项目基金(60496320,60496321),国家自然科学基金(60773097,60873148),新世纪优秀人才支持计划项目基金,吉林省科技发展计划项目基金(20060532,20080107)和欧盟项目基金TH/Asia Link/010(111084),吉林省科技发展计划项目(No. 20060532)资助。

王晓宇(1984—),女,硕士生,主要研究方向为基于模型的诊断;欧阳丹彤(1968—),女,教授,博士生导师,主要研究方向为基于模型的诊断、自动推理和模型检测,E-mail:ouyangdantong@163.com;赵相福(1981—),男,博士生,主要研究方向为基于模型的诊断;常晓环(1984—),女,硕士生,主要研究方向为时空推理等。

定义 2(前集[9]) 设 $x,y\in X$ 为 Petri 网 $N=(S,T;F)$ 中任意一个元素，则 x 的前集表示为 $^{*}x$，定义为 $^{*}x=\{y|(y,x)\in F\}$[9]。

定义 3(后集[9]) 设 $x,y\in X$ 为 Petri 网 $N=(S,T;F)$ 中任意一个元素，则 x 的后集表示为 x^{*}，定义为 $x^{*}=\{y|(x,y)\in F\}$[9]。

定义 4(token[9]) token 是库所中的动态对象，可以从一个库所移动到另一个库所。

1.2 离散事件系统诊断基础

定义 5(部件模型) 一个部件模型可由一个四元组来表示：$D=(S,T;F,\text{token})$。

其中，S 是状态集合，表示部件的所有可能状态，包括故障状态；T 是部件的事件集合，表示触发系统状态改变的那些事件；F 是 $S\times T\cup T\times S$ 的子集；token 是令牌，动态的标志，表示当前系统的状态。

定义 6(同步) 将子系统模型通过通讯事件的发生，在时间上进行一致化，对模型进行合并的操作。

定义 7(token) 某 token 当前所处状态的位置。

1.3 诊断系统建模

利用一些分布式系统的性质[2]，给出模型的建立规则：

- 部件模型之间独立，有严格的边界，仅通过通讯事件联系。
- 模型之间的通讯和模型内部的通讯都是异步的。
- 部件本身可以提供相关信息：本地故障或者是需要同步。

2 算法描述

2.1 同步

构建系统时，需要为每个部件建立 Petri 网模型，部件的状态 $s\in S$，事件 $t\in T$。系统运行到故障事件出现时，需要对这些部件进行诊断。由于系统行为的复杂性，故障涉及的部件有时会多于一个，此时单个的部件模型不能满足诊断的需要，需要将多个部件模型同步，建立起一个综合模型，在这个综合模型上进行诊断，得出故障原因。因此，模型的同步直接关系到诊断结果的得出，同时模型同步的方法和效率对于诊断的效率也有较大影响。

离散事件系统部件之间的同步是建立在各模型的通讯事件上的，T 是系统的事件集合。在 T 中包括以下几个部分：正常事件 Ni、故障事件 Fi、观测事件 Oi、各部件之间进行相互联系的通讯事件 Ci。每个事件对系统有着不同的作用，并且各事件集合之间是相互独立的，即 $Ni\cap Fi\cap Oi\cap Ci=\varnothing$[1]。

正常事件和故障事件是部件自有的。通讯事件一旦发生，就至少有两个部件参与。观测事件是我们对系统运行时的情况进行观测。观测和系统模型的同步机进行同步乘积得到的就是诊断结果，或者可以从得到的同步机中提取诊断的轨迹。所以，将部件同步的时候，首先考虑的就是以通讯事件为关键路径，建立同步机。

为了方便表示 Petri 网中的节点的顺序关系，用—>来表示节点顺序。

算法 1 同步

输入：M1={S1,T1;F1,token1},M2={S2,T2;F2,token2}

输出：M={S,T;F,token}

1. 初始化：S′=∅,T′=∅,T0=∅,F′=∅
2. 令 T0=T1∩T2
3. 令 *C1={*t1| t1∈T0,*t1∈S1}
   C1*={t1*|t1∈T0,t1*∈S1}
   *C2={*t2| t2∈T0,*t2∈S2}
   C2*={t2*|t2∈T0,t2*∈S2}
4. forall:t0∈T0
   &&((*t0—>t0—>t0*)∈F1)
   &&((*t0—>t0—>t0*)∈F2))
   t1=t0,*t1∈*C1,t1*∈C1*
   t2=t0,*t2∈*C2,t2*∈C2*。
   F′=F′∪{(*t1×*t2)—>t0—>(t1*×t2*)}
   S′=S′∪{*t1×*t2}∪{t1*×t2*}
5. forall:
   if &token1×&token2∈S′
   &&((&token1)*∉T0||(&token2)*∉T0
   F′=F′∪{(&token1×&token2)—>(&token1)*
   —>(((&token1)*)*×&token2)}∪
   {(&token1×&token2)—>(&token2)*
   —>(&token1×(&token2)*)*}
   S′=S′∪{(((&token1)*)*×&token2}∪{&token1×((&token2)*)*}
6. if (未到达终止节点)
   forall:(s1×s2∈S′)&&
   (s1*∉T0)&&(s2*∉T0)
   F′=F′∪{(s1×s2)—>s1*—>((s1)*)*×s2}∪
   {(s1×s2)—>s2*—> s1×((s2)*)*}
   S′=S∪{((s1)*)*×s2}∪{s1×((s2)*)*}
7. if &token1×&token2∉S′
   forall:(token1*∉T0))&&(token2*∉T0)
   F′=F′∪{(token1×token2))—>token1*
   —>((token1)*)*×token2}∪
   {(token1×token2)—>token2*
   —> token1×((token2)*)*}
   S′=S′∪{(((&token1)*)*×&token2}∪
   {&token1×((&token2)*)*}
8. S=S′,T=T1∪T2,F=F′,token=token1×token2.

我们将就此算法给出一个简单的实例，图 2 是实例的 Petri 网模型。

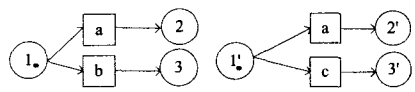

图 2 Petri 网建模自动机

初始集合如下：

S1={1,2,3},S2={1′,2′,3′}

E1={a,c},E2={a,b}

T1={(1,a,2),(1,c,,3)},T2={(1′,a,2′),(1′,b,3′)}

token1=1,token2=1′

下面开始执行算法：

1. 初始化：S′=∅,T′=∅,T0=∅,F′=∅
2. E0={a}
3. *C1={1},C1*={2},*C2={1′},C2*={2′}
4. F′={(11′—>a—>22′)}
   S′={11′,22′}
5. F′={(11′—>a—>22′),(11′—>c —>31′),
   (11′—>b—>13′)}
   S′={11′,22′,31′,13′}
6. F′={(11′—>a—>22′),(11′—>c —>31′), (11′—>b—>13′),(31′—>b—>33′),(13′—>c —>33′)}.
   S′={11′,22′,31′,13′,33′}
7. F′={(11′—>a—>22′),(11′—>c —>31′),

(11′—>b—>13′),(31′—>b—>33′)
(13′—>c —>33′)}
S′={11′,22′,31′,13′,33′}
8. S={11′,22′,31′,13′,33′}.
T={a,b,c}
F={(11′—>a—>22′),(11′—>c —>31′),
(11′—>b—>13′),(31′—>b—>33′),(13′—>c —>
33′)}
token=11′

图 3 给出的是执行算法之后得出的 Petri 网模型。

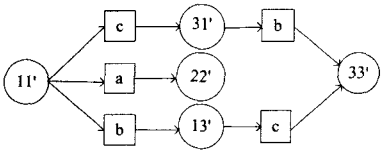

图 3 同步自动机

2.2 诊断

在本文提出的诊断过程中，给出一种分布式的诊断方法，由每个部件进行局部诊断，不能处理的诊断就将两个部件进行同步，形成一个更大的同步机，作为一个模型进行诊断。下面对必要的过程做些解释。

&token 的作用是定位系统当前所处的状态。Petri 网是基于异步信息流的离散并行表示模型，诊断系统运行时，token 也在 Petri 网中流动。根据系统的初始状态，有一个或者多个 token。系统开始运行时，token 在状态和事件之间流动。

算法 2 诊断

diagnosis(model$_i$)

1. 初始化：token0=current(state)。token$_i$=model$_i$(token)。

//current(state)表示当前的状态，token$_i$ 表示的是第 i 个模型的令牌。

2. while (((token$_i$)! =token$_0$)
&&((token$_i$)! =end)))

//end 表示模型定义时给出的终止状态。

movelocal(token$_i$)

//在本地模型中进行令牌的传递，进行诊断。

3. if (token$_i$=token$_0$)
Store(path)

//存储当前的 token 的轨迹。

4. else
Syn(model$_1$,model$_2$)

//按照算法 1 将两个模型同步起来。

5. diagnosis(SynModel)

//在同步模型中进行诊断。

3 分布式诊断模型的几个性质

3.1 独立性

定义 8(状态独立性[10]) 令 S_i 和 S_j 是 Model$_i$ 和 Model$_j$ 的状态集合，$S_i \cap S_j=\emptyset$，$S_i^* \cap S_j^*=\emptyset$，$S_i^* \cap S_j^*=\emptyset$，$i\neq j$，那么说这两个状态就是独立的。也就是说如果一个状态没有和其他部件状态同步的事件发出的话，这个状态就是独立的。

定义 9(转换独立性[10]) 令 T_i 和 T_j 是 Model$_i$ 和 Model$_j$ 的转换集合，如果 $T_i \cap T_j=\emptyset$，$i\neq j$，那么称 T_i 和 T_j 是转换独立的。

如果事件的状态独立，可以很容易求出两个子模型的同步机，只需要做一个笛卡尔乘积即可。但是这个同步机占用的空间很大，并且如果两个模型之间没有交互事件，则说明它们之间没有相互作用，是相对不太可能互相影响的部件。同步之后，作为一个模型能够检测出故障的可能性也是很小的。但是对于整个系统来说，研究部件状态之间的独立性是很有意义的。若能证明两个状态之间是状态独立的，就可以选择其他的部件进行同步。这在同步时间的复杂性上是有些好处的。

事件的转移独立对于整个系统的诊断空间有很大的影响。整个的同步机就是以同步的事件为基础骨架建立起来的。如果所有的转移都是独立的，即算法 1 中的第 2 步，交集将为空，那么两个模型无法进行同步，也就无法形成一个较大的同步机。如果只有一部分转移是独立的，那么可以用不独立的那些转移进行同步，而将这些独立的转移或者作为同步机中的一部分，或者因为无法到达最终状态而剪枝。

3.2 可行性分析

本文提出的算法是可行的。下面对算法的可行性进行讨论，说明诊断可以完全给出，或者在有限的时间内结束。

首先，如果部件自身能够进行诊断，则返回诊断结果。

其次，如果部件自身的诊断不能解释故障发生的原因，则进行同步，继续诊断。

最后，若同步结束，没有诊断，则算法将在同步结束的同时结束。

3.3 复杂性

对于各自部件的诊断来说，诊断的时间复杂度是 $O(n_1+n_2)$，其中 n_1 是状态的个数，n_2 是转换的个数。诊断的时间应该是线性的。

同步过程的时间复杂性为 $O(n_1\times n_2)$，构造整个同步机时间复杂性是 $O(n_1\times n_2\times m)$，其中 m 是部件个数。空间复杂性为 $O(m\times n)$，n 是每个部件的平均节点数。

3.4 与其它诊断方法的比较

离散事件系统中最具有代表性的是诊断器理论[11,12]。该理论使用正则语言对系统行为建模。主要思想是：一个语言是可诊断的，是指在有限的延迟内能够探测到某些可区分的不可观测的事件，即故障事件。诊断器用系统行为的在线观测来进行诊断，同时在离线时对可诊断性的充分条件和必要条件进行规定和校验。这种诊断器的方法提供了相当快的在线诊断速度，但是需要很大的空间来存储预先计算出来的结果和诊断事件的过程轨迹。

文献[2]中的算法减小了诊断所需的空间，但是速度有所降低。该方法在计算诊断的时候，通过分析诊断过程中出现的故障事件的顺序，推论出故障出现的原因。这种方法的缺点在于子系统同步时，每个故障发生都会产生一个副本，空间呈指数级上升。

本文提出的算法在诊断空间上有所降低，只需要存储部件的模型的有限空间和计算诊断时不超过部件模型节点总个数的空间即可。而分布式计算使得部件的诊断可以同步进行，同步诊断，从而减少计算的时间。

4 实验结果

表 1 是在 VC++6.0 下编程实现算法 1 的结果，表中的时间是算法 1 运行有该行节点数的两个模型，在有该列同步事件数的条件下所需时间。由此可知，本算法有着很好的时间效果。并且，同步事件与节点数的比例达到一定程度的条件下，算法的时间效率会更高。

(下转第 199 页)

这部分实验主要考察迁移代频对并行算法的影响，变量 s 表示迁移代频，变量 *generation* 表示演化代数，这里 *generation*=5000，参数设置如表 5 所列。

表 5　参数设置

S-1	s=50
S-2	s=100
S-3	s=200
S-4	s=500

由图 6、图 7 知，迁移代频越大，即每次迁移相隔的代数越长，则无论对同步并行算法还是异步并行算法来说，花费的运行时间都越少；对于 AF，同步并行算法和异步并行算法都是在迁移代频 s=200 时达到最好，因此迁移代频不宜设置太频繁。

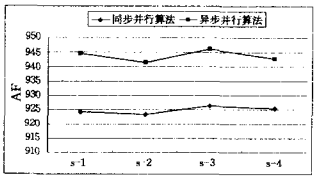

图 6　迁移代频对平均适应值的影响

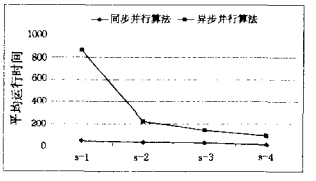

图 7　迁移代频对平均运行时间的影响

结束语　本文将 EDA 和多种群策略引入 GEP 中，提出了基于 GEGEP 的同步及异步并行算法。此并行算法提高了搜索能力，减少了计算时间，取得了较好的效果。

(1)EDA 是一种很好的学习算法，有利于提高收敛速度，多种群策略有利于跳出局部最优，个体迁移策略有利于增加种群多样性；

(2)并行算法与串行算法相比，并行算法获得了较好的解；

(3)同步并行算法和异步并行算法相比，同步并行算法获得了线性加速比，具有较好的解。异步并行算法虽没有获得线性加速比，但平均适应值、最好适应值和拟合误差的结果都优于同步并行算法。然而，同步并行算法的运行时间明显优于异步并行算法。

参 考 文 献

[1] Ferreira C. Gene Expression Programming: A New Adaptive Algorithm for Solving Problems[J]. Complex Systems, 2001, 13(2): 87-129

[2] Holland J H. Adaptation in Natural and Artificial Systems[M]. Cambridge: MIT Press, 1992

[3] Koza J R. Genetic Programming: On the Programming of Computers by Natural Selection[M]. Cambridge: MIT Press, 1992

[4] 杜欣，刘坤起，康立山，等. M-GEP-GA：基于多层染色体基因表达式程序设计的混合遗传进化算法[J]. 计算机应用，2007，27(4)：956-959

[5] 谢大同，康立山，李悦乔，等. 符号回归的一种新算法[J]. 系统仿真学报，2007，8(19)：1667-1671

[6] 杜欣，康立山，李悦乔，等. 基因评估基因表达式程序设计方法[J]. 小型微型计算机系统，2007，5(28)：834-840

[7] 曹宏庆，康立山，陈毓屏，等. 常微分方程组并行演化建模的实验研究[J]. 软件学报，2003，3(14)：443-450

[8] 管宇，徐宝文. 基于模式迁移策略的并行遗传算法[J]. 计算机学报，2003，26(3)：294-301

[9] Ferreira C. Gene expression programming: A new adaptive algorithm for solving problems[J]. Complex Systems, 2001, 13(2): 87-129

[10] Ferreira C. Mutation, transposition, and recombination: An analysis of the evolutionary dynamics[C]// Proceedings of the 6th Joint Conference on Information Sciences. USA, 2002

[11] 彭京，唐常杰，李川，等. M-GEP：基于多层染色体基因表达式编程的遗传进化算法[J]. 计算机学报，2005，9(28)：1459-1466

[12] 段磊，唐常杰，左劼，等. 基于基因表达式编程的抗噪声数据的函数挖掘方法[J]. 计算机研究与发展，2004，41(10)：1684-1689

[13] 蒋思伟，蔡之华，曾丹，等. 基于模拟退火的并行基因表达式算法研究[J]. 电子学报，2005，33(11)：2017-2021

[14] 杜欣，刘坤起，陈玉军. 改进的基因表达式程序设计实现复杂函数的自动建模[J]. 微计算机信息，2006(9)

[15] Larranaga P, Lozano J A. Estimation of distribution algorithms: A new tool for evolutionary computation[M]. Kluwer Academic Publishers, 2001

[16] 都志辉. 高性能计算并行编程技术-MPI 并行程序设计[M]. 北京：清华大学出版社，2001

(上接第 182 页)

表 1　模型同步时间表

节点个数		同步事件数			
		2	5	10	20
12	8	<1ms	<1ms	—	—
20	30	34ms	46ms	62ms	—
50	50	453ms	469ms	281ms	297ms
80	120	488ms	503ms	329ms	312ms

结束语　本文以 Petri 网为建模工具，对离散事件系统进行诊断，提出同步扩展模型的方法，解决了建模规模和诊断完备之间的冲突。通过实验分别得出了同步事件数与诊断节点数对同步过程的影响。

参 考 文 献

[1] Yannick P. Diagnosability analysis of distributed discrete event systems[C]// European Conference on Artificial Intelligence. Valencia, Spain, 2002

[2] Fabre E, Benveniste A, Haar S, et al. Distributed Monitoring of Concurrent and Asynchronous Systems[J]. Discrete Event Dynamic Systems, 2005, 15(1): 33

[3] Anika S, Yannick P. Scalable Diagnosability Checking of Event-driven Systems[C]// JCAI. Hyderabad, India, 2007

[4] Jussi R. Diagnosers and Diagnosability of Succinct Transition Systems[C]// IJCAI. Hyderabad, India, 2007

[5] Console Luca, Picardi Claudia, Dupre D T. A Framework for Decentralized Qualitative Model-Based Diagnosis[C]// IJCAI. Hyderabad, India

[6] Jiroveanu G, Boel K R. Petri Net model-based distributed diagnosis for large interacting systems [C] // 16th International Workshop on Principles of Diagnosis. Pacific Grove, CA, USA, 2005

[7] Petri C A. Fundamentals of a theory of asynchronous information flow[C]// IFIP Congress. Amsterdam, North Holland, 1963

[8] Cassez F, Roux O H. From Time Petri Nets to Timed Automata [J]. Journal of Systems and Software, 2006, 79(10): 1456

[9] 袁崇义. 佩特里网(1982 年第一版)[M]. 南京：东南大学出版社，1989：239

[10] Cordier M O, Grastien A. Exploring independence in a decentralized Approach of Diagnosis[C]// IJCAI. Hyderabad, India, 2007

[11] Sampath M, Sengupta R, Lafortune S, et al. Diagnosability of Discrete-event Systems[J]. Automatic Control, 1995, 40(9): 1555

[12] Sampath M, Sengupta R, Lafortune S, et al. Failure Diagnosis Using Discrete-event Models[J]. Control Systems Technology, 1996, 4(2): 105, 2007