第 38 卷　第 11 期
Vol.38　No.11

计　算　机　工　程
Computer Engineering

2012 年 6 月
June 2012

·工程应用技术与实现·　文章编号：1000—3428(2012)11—0228—03　文献标识码：A　中图分类号：TN912

基于故障树的专家系统推理机设计

陈　正[1,2]，李华旺[1,2]，常　亮[1,2]

(1. 上海微小卫星工程中心，上海 200050；2. 中国科学院上海微系统与信息技术研究所，上海 200050)

摘　要：针对微小卫星强实时性和资源受限的特点，提出一种基于故障树的专家系统推理机。根据广度优先搜索设计正向推理算法，根据深度优先搜索设计逆向推理算法，2 种算法在时间和空间上均满足线性复杂度。实验结果表明，该推理机可满足微小卫星对实时性的要求，同时也能节省星上资源。

关键词：推理机；专家系统；故障树；故障诊断；图算法；微小卫星

Design of Inference Engine for Expert System Based on Fault Tree

CHEN Zheng[1,2], LI Hua-wang[1,2], CHANG Liang[1,2]

(1. Shanghai Engineering Center for Micro-satellite, Shanghai 200050, China;
2. Shanghai Institute of Micro-system and Information Technology, Chinese Academy of Sciences, Shanghai 200050, China)

【Abstract】According to the rigorous requirements for real-time performance and conditional resource of the micro-satellite, inference engine of expert system is designed based on fault tree. It designs forward direction inference algorithm based on breadth-first search, and reverse direction inference algorithm based on depth-first search search. Time complexity and space complexity of two algorithms are both linear. Experimental result shows that the engine can not only improve the efficiency of inference engine, but also save the resource of the micro-satellite.

【Key words】inference engine; expert system; fault tree; fault diagnosis; graph algorithms; micro-satellite

DOI: 10.3969/j.issn.1000-3428.2012.11.070

1　概述

高可靠性是现代微小卫星的一个显著特点，这要求微小卫星必须具备的能力之一就是自主故障诊断[1-3]。基于专家系统的故障诊断技术近年来在航天领域得到广泛的应用，作为专家系统核心组成之一的推理机，其性能直接影响到专家系统的结果和性能[3]。目前国内关于专家系统推理机的研究并未对时间和空间进行综合的考虑[3-5]，而实时性和资源有限性是微小卫星显著的特点，因此，本文综合考虑微小卫星专家系统推理机的时间和空间约束，在基于故障树的专家系统中，结合图搜索的相关理论，提出基于广度优先搜索的正向推理算法和基于深度优先搜索的逆向推理算法。

2　专家系统推理原理分析

专家系统的推理方式和分类方法较多[3]。按推理时所用的知识确定，推理可分为确定性推理和不确定性推理；按推理过程中所推出的结论是否越来越接近最终目标来分类，可分为单调推理和非单调推理；按推理的方向来分类，又可以分为正向推理和逆向推理。本文即按推理方向分类方法对推理机进行研究。

2.1　正向推理

正向推理又称为数据驱动或前件推理，其基本原理是：从问题已有的事实出发，正向使用规则，当规则的条件部分与已有的事实匹配时，就把该规则作为可用规则放入候选规则队列中，然后通过冲突消解，在候选队列中选择一条规则作为启动规则进行下一步推理。重复该过程，直到再无可用规则或求得所要求的结果为止。图 1 为专家系统正向推理的流程。

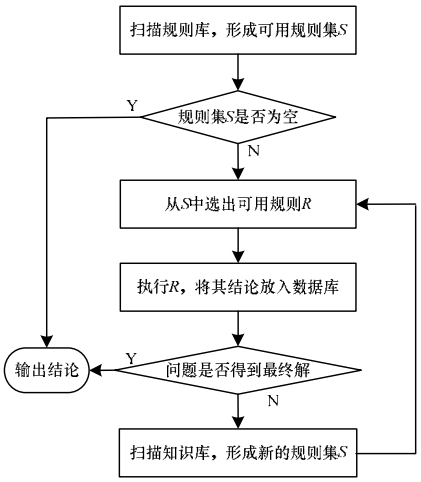


图 1　专家系统正向推理流程

2.2　逆向推理

逆向推理又称为目标驱动或后件推理，其推理过程与正向推理正相反，首先提出某个假设，然后寻找支持该假设的证据，若所需的证据都能找到，说明假设是正确的，否则说明假设不成立，此时需另作假设，直到找到支持假设成立的证据。图 2 为专家系统逆向推理的流程。

基金项目：上海市科委专项基金资助项目(10DZ2291700)

作者简介：陈　正(1986－)，男，硕士研究生，主研方向：卫星自主健康管理，星载软件设计；李华旺，研究员、博士；常　亮，副研究员、硕士

收稿日期：2011-08-22　**E-mail：**chenzheng1030@gmail.com

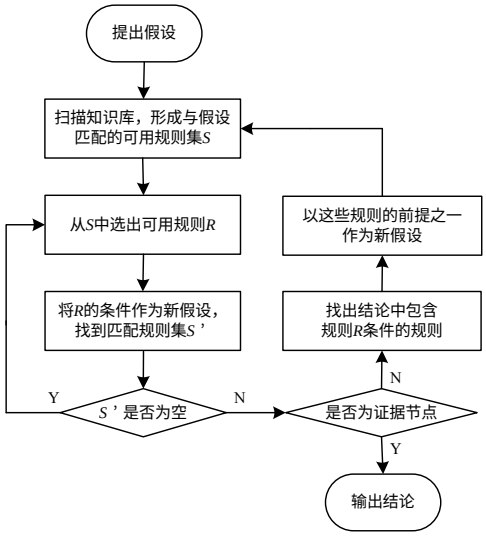


图 2　专家系统逆向推理流程

2.3　正向推理与逆向推理的比较

表 1 给出了正向推理与逆向推理的特点比较。

表 1　正向推理与逆向推理比较

特点	正向推理	逆向推理
演绎推理	事实→目标	目标→事实
初始条件	初始事实集	目标和初始事实集
结束条件	包含所有目标节点	以满足条件的目标事实节点作为终节点
优点	直观，允许用户提供事实信息	不必使用与目标无关的规则，且利于向用户提供解释
缺点	推理目标不明确，求解问题要执行许多与解无关的操作，推理时可能发生组合爆炸	高度依赖提出的初始假设

3　基于故障树的推理机设计

故障树鲜明的层级结构以及其易维护、易拓展的特性使得基于故障树的专家系统非常适合于结构复杂、故障类型繁多的微小卫星故障诊断领域[6-7]。

3.1　故障树的存储

故障树主要包括事件(event)和逻辑门(logic_gate)2 个部分，事件又可分为顶事件(top event)、中间事件(middle event)和基本事件(base event)。故障树表示的知识在逻辑上形成一个复杂的广义有向图 $G=(V, E)$，其中，V 为顶点，包括事件和逻辑门 2 种；E 为有向弧。故障树的结构存储形式直接影响到推理效率和搜索速度。有向图通常采用邻接表表示法或者邻接矩阵表示法[7]。考虑到需要动态的在图中增加、删除节点，并考虑到整个图需要进行正向和逆向的推理，本文使用邻接表表示法。节点的结构如下(使用 C/C++描述)：

```
typedef enum NodeType{ EVENT, LOGIC }; //E 代表事件，L 代
//表逻辑门
typedef  enum  Color{ WHITE, GRAY, BLACK };
//WHITE：节点未访问，GRAY：访问过但不存在故障，BLACK：
//存在故障
typedef  enum  LogicType{AND,OR};
//AND 表示与门，OR 表示或门
typedef  struct
{
    vector<int>     previous;          //前驱列表，若 previous
//为空，则表示该节点为 base event
    vector<int>     post;          //后继列表，若 post 为空，则表
//示该节点为 top event
    int     num;          //节点编号，给前驱、后驱提供索引
NodeType     nodetype;          //节点类型
union
{
 LogicType   logictype;      //逻辑门类型
 char        *descrip;      //对事件节点的描述
}node_description;
Color          color;          //用于表示节点访问情况
}TreeNode;
```

从该类的定义可以看出，事件节点的前驱与后继若存在，则必为逻辑节点；同理逻辑节点的前驱与后继也必为事件节点。整个故障树的信息只需要储存在一个 vector 内即可，即：

```
vector<TreeNode>          FaultTree;
```

3.2　广度优先搜索在正向推理机中的应用

正向推理的思想是从基本事件出发，一层一层向顶事件扫描，最终得出顶事件是否发生的结论。而图的广度优先算法正是借助于队列，一层一层往下扫描[8]。因此，正向推理机借用图算法的广度优先算法进行推理。算法的输入为初始故障基本事件集合 *P_SET* 和故障树，定义队列 *OPEN* 表示待处理节点列表，输出为判定故障树的顶事件是否发生，算法的具体代码如下：

```
Forward_Reasoning(P_SET, FaultTree)
OPEN ← P_SET
初始化 E_Flag 和 L_Flag 为 WHITE
for i ← 0 to length[P_SET]
    do E_Flag[P_SET[i].num] ← BLACK
while OPEN  非空
    do E_temp ←  OPEN 队首元素
      if E_temp 为顶事件
       then break
      for i ← 0 to length[E_temp.post]
        do L_temp ← E_temp.post[i]
          if L_Flag[L_temp.num] = WHITE
           then L_Flag[L_temp.num] = GRAY
            if CHECK-FIRE(L_temp) =true
             then L_Flag[L_temp.num] = BLACK
               OPEN ← L_temp.post
    将 E_temp 从 OPEN 中去除
if OPEN 非空
 then  发现系统故障
else  未发现系统故障，仅存在部件级故障
```

通过分析可知，该算法的时间复杂性为 $O(V+E)$，空间复杂性为 $O(V)$，其中，V 为故障树顶点；E 为故障树有向弧。

3.3　深度优先搜索在逆向推理机中的应用

逆向推理的主要思想是从顶事件出发，以此检查每一条可能导致顶事件发生的路径，最终确定导致顶事件发生的基本事件。图算法中的深度优先搜索正是利用栈，每次搜索一条支路，然后通过栈操作重新选择另外一条支路，直到寻找到搜寻的结果[8]。因此，逆向推理机借助图算法的深度优先搜索算法进行推理。算法包括 2 个部分：

(1)主函数，输入为检测到到的故障节点 p 和故障树，输出为栈 S，S 内的元素为引起故障节点 p 的故障基本事件集合。

(2)深度搜索故障树函数，输入为需要检测的节点 p 和初始故障基本事件集合 *P_SET*，输出为检测的节点的结果。

算法的具体代码如下：

```
Backward_Reasoning(p, FaultTree)
初始化所有节点的 Color ← WHITE
time ← 0
栈  S ← ∅
TreeDFS(p, P_SET)
Color TreeDFS(p, P_SET)
```

```
if p.color ≠ WHITE       //该节点已经访问过
    return p.color
if p.previous = NIL && p.NodeType = EVENT
                         //该事件为基本事件
    if p ∈ P_SET
        p.color ← BLACK
        time ← time+1
        push(S,{p.num,time})
        return BLACK
    else
        p.color ← GRAY
        return GRAY
if p.previous ≠ NIL && p.NodeType = EVENT
                         //该事件为中间事件或顶事件
    t = p.previous
        if TreeDFS(t, P_SET) = GRAY
            p.color ← GRAY
            return GRAY
        else
            p.color ← BLACK
            return BLACK
else tmptime = time      //该节点为逻辑节点
    if p.LogicType = AND
        for each TreeNode e ∈ p.previous
            if TreeDFS(e, P_SET) = GRAY
                while top(S).time ≥ tmptime
                    pop(S)
                p.color ← GRAY
                return GRAY
        p.color ← BLACK
        return BLACK
    else
        for each TreeNode e ∈ p.previous
            if TreeDFS(e, P_SET) = BLACK
                p.color ← BLACK
                return BLACK
        while top(S).time ≥ tmptime
                pop(S)
        p.color ← GRAY
        return GRAY
```

通过分析可知，该算法的时间复杂性为 $O(V+E)$，空间复杂性为 $O(V)$。

4 算法验证

图 3 为某微小卫星姿控系统的故障树，其中，深色的代表初始故障基本事件集合 $P\_SET=\{1, 3, 5, 6, 9\}$，顶事件故障在逆向推理中使用。

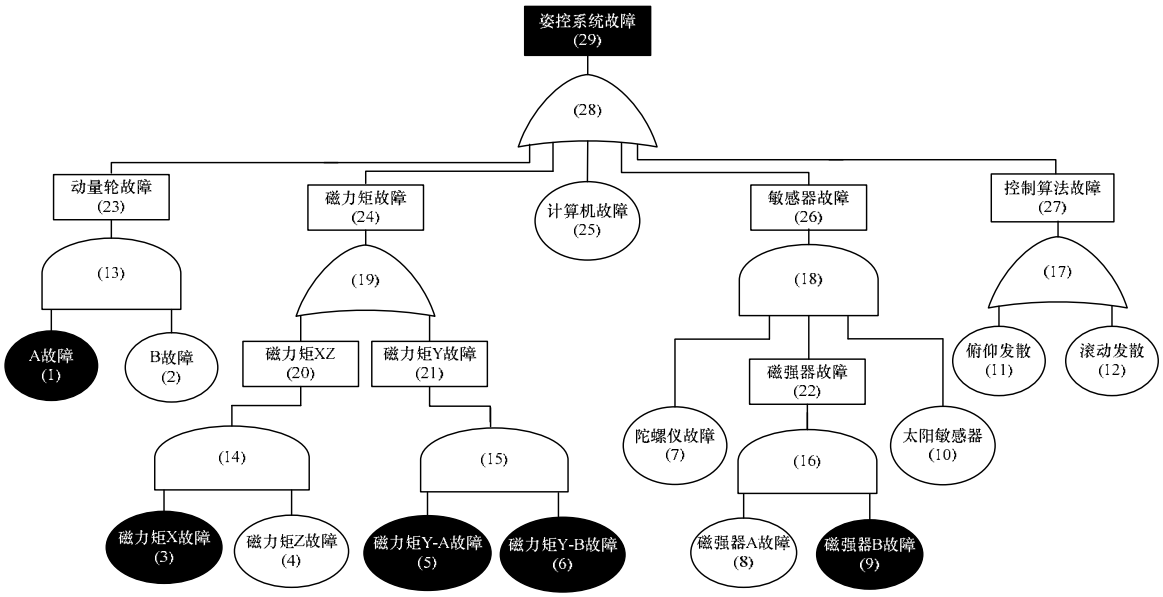
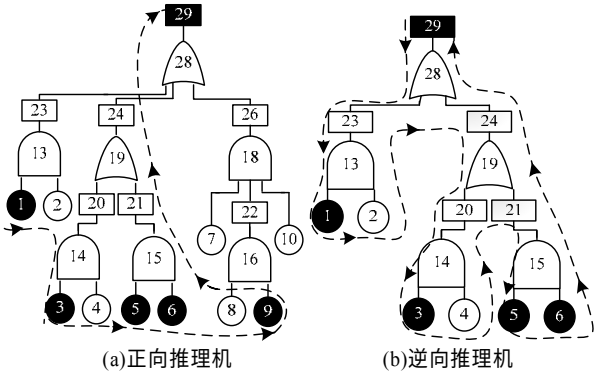


图 3 某卫星姿控系统故障树

图 4(a)为基于广度优先搜索的正向推理机的推理过程，图 4(b)为基于深度优先搜索的逆向推理机的推理过程，其中，2 个白色点表示推理过程中使用的中间节点。



(a)正向推理机 (b)逆向推理机

图 4 推理机的推理过程

在 VS2005 中对推理机进行仿真，结果如图 5 所示。可以看出，正向推理与逆向推理算法都符合设计要求。

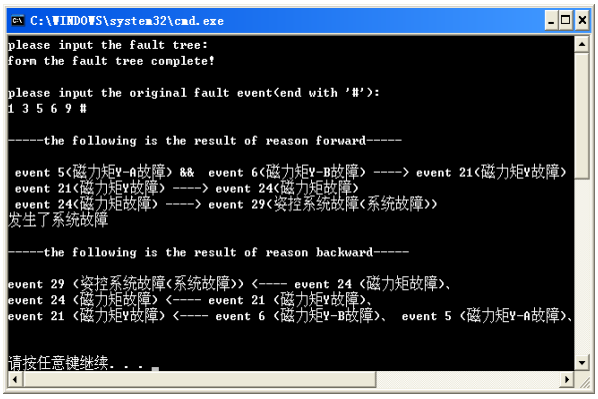


图 5 推理机的仿真结果

(下转第 250 页)

问是下订单)，从图 5 可以看到当浏览用户数量小于 45 时，Full Replication 的性能最好。这是因为 TPC-W 基准负载刚开始呈现较低的时间局部性，所以内容无关缓存的效果不明显。但是随着浏览用户不断增加，用户的访问呈现较高的时间局部性，此时内容无关缓存作用凸显，而 Full Replication 所有查询都需要通过数据库服务器处理，内部处理时延增加，因此，ASDG 和 GlobeCBC 系统性能要比 Full Replication 好。当用户数量大于 50 后，GlobeCBC 时延呈现缓慢下降，同时 ASDG 性能趋于稳定，所以在未来可见的负载数量状态下，GlobeCBC 与 ASDG 的浏览模式性能将相差不大。

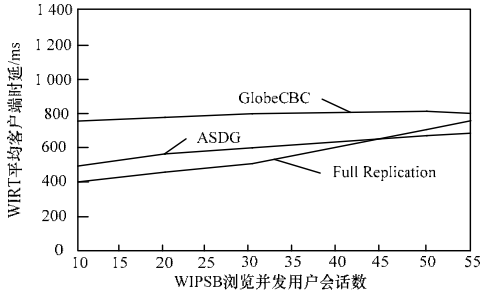


图 5 浏览模式下系统性能对比

订购模式(用户对 50%的网页访问是浏览，50%的网页访问是下订单)，由于订购模式包含了大量的更新操作，而 GlobeCBC 只适合于读多更新少的应用，因此 ASDG 和 Full Replication 的 WIRT 要比 GlobeCBC 低，如图 6 所示。

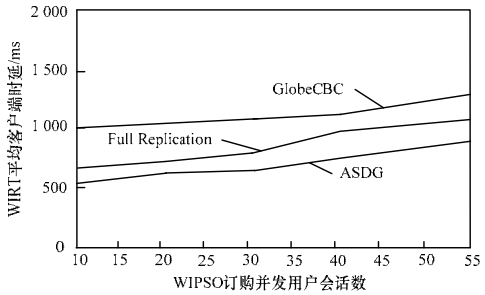


图 6 订购模式下系统性能对比

对于 Full Replication 而言，由于所有的更新操作都需要转交给 OS，而且为了维护数据的一致性，用户对数据库的写操作需要在所有的数据库服务器上执行一遍，这会增加额外的网络流量和服务器的处理开销，导致了时延增大。然而，ASDG 采用了部分复制策略，正如前文所说，ASDG 将一部分请求转交给了相关主服务器来处理，而且更新仅仅在一些服务器上执行，降低了完全复制的开销。因此，在订购模式下，ASDG 的性能最好。

6 结束语

本文总结前人在动态数据缓存方面的工作并分析了各自的优缺点，提出缓存与部分复制的系统方案 ASDG，通过 TPC-W 基准测试表明：ASDG 在读写 2 种请求模式下都表现出较好的性能优势，满足了多种动态 Web 应用加速的需求。

下一步将解决当用户的访问形式随着时间而改变时，系统需要执行重聚才能支持好的性能，但执行重聚的开销太大的问题。

参考文献

[1] Challenger J, Dantzig P, Witting K. A Fragment-based Approach for Efficiently Creating Dynamic Web Content[J]. ACM Transactions on Internet Technology, 2005, 5(2): 359-389.

[2] Amiri K, Park S, Tewari R, et al. DBProxy: A Dynamic Data Cache for Web Applications[C]//Proc. of International Conference on Data Engineering. [S. l.]: IEEE Press, 2003: 821-831.

[3] Sivasubramanian S, Pierre G, Steen M, et al. GlobeCBC: Content-blind Result Caching for Dynamic Web Applications[R]. Vrije Universiteit, Technical Report: IR-CS-022, 2006.

[4] Cecchet E. C-JDBC: A Middleware Framework for Database Clustering[J]. Data Engineering, 2004, 27(2):19-26.

[5] Sivasubramanian S, Pierre G, Steen M. GlobeDB: Autonomic Data Replication for Web Applications[C]//Proc. of International Conference on World Wide Web. Chiba, Japan: [s. n.], 2005: 33-42.

[6] Pierre G, Steen M, Tanenbaum A S. Dynamically Selecting Optimal Distribution Strategies for Web Documents[J]. IEEE Trans. on Computers, 2002, 51(6): 637-651.

[7] Groothuyse T, Sivasubramanian S, Pierre G. GlobeTP: Template-based Database Replication for Scalable Web Applications[M]. Alberta, Canada: ACM Press, 2007.

[8] National Institute of Standards and Technology. NIST Net[EB/OL]. (2005-07-20). http://snad.ncsl.nist.gov/itg/nistnet/.

编辑 顾逸斐

(上接第 230 页)

5 结束语

本文提出基于故障树的专家系统推理机，将图搜索算法的广度优先遍历算法应用于正向推理，深度优先遍历算法应用于逆向推理。推理算法在时间和空间上都满足线性复杂度，不仅能够满足微小卫星对实时性的要求，同时线性的空间需求使该算法也为微小卫星节省了存储资源。本文设计适用于结构复杂的应用领域，对资源受限和要求强实时性的应用场合也有借鉴意义。但该设计只能诊断单一故障，因此，下一步将设计图算法推理机用于诊断多故障。

参考文献

[1] 姜连祥，李华旺，杨根庆，等. 航天器自主故障诊断技术研究进展[J]. 宇航学报, 2009, 30(4): 1320-1326.

[2] Schetter T, Campell M, Surka D. Multiple Agent-based Autonomy for Satellite Constellations[J]. Artificial Intelligence Research, 2003, 145(1/2): 147-180.

[3] 王亚南. 专家系统中推理机制的研究与应用[D]. 武汉: 武汉理工大学, 2006.

[4] Chen Wenbin, Liu Xiaoling, Fang Yu, et al. Inference Engine Design of Expert System Based on Blackboard Model and Fault Tree[C]//Proc. of Asia-Pacific Conference on Information. Washington D. C., USA: IEEE Press, 2009: 18-20.

[5] 曲朝阳，高宇峰，聂 欣. 基于决策树的网络故障诊断专家系统模型[J]. 计算机工程, 2008, 34(22): 215-217.

[6] 张春华，刘 伟. 基于故障树的故障诊断专家系统[J]. 兵工自动化, 2009, 28(11): 15-17.

[7] 张 键，廖 瑛，庄景钊. 基于故障树分析法的某型直升机故障诊断专家系统设计分析[J]. 航空计算技术, 2002, 32(3): 85-87.

[8] Cormen T H, Leiserson C E, Rivest R L, et al. Introduction to Algorithms[M]. 2nd ed. Boston, USA: Massachusetts Institute of Technology Press, 2001.

编辑 金胡考