基于 Vortex 工作流的软件应用研究

杨 俊 赵晓琴 夏宽理

(复旦大学计算机科学系软件工程实验室 上海 200433)

摘 要 Vortex 工作流以属性为中心的特性,为适应系统设计对任务与数据的不同侧重提供了很大的灵活性。通过引入说明式规范与应用层描述语言,Vortex 工作流具有良好的可理解性与可修改性,这对工作流管理系统的推广应用具有重大意义。本文主要介绍 Vortex 工作流的各类基本要素及概念,并就某些相关问题作定性讨论,最后结合一个应用实例按 Vortex 规范进行建模分析。

关键词 模式 快照 启动条件 边际效应 联合函数 联合语义语言

A RESEARCH OF VORTEX WORKFLOW - BASED SOFTWARE APPLICATION

Yang Jun Zhao Xiaoqing Xia Kuanli

(*Department of Computer Science, Fudan University, Shanghai 200433*)

Abstract Regarding the attribute as centric, the character held by Vortex workflow provides vast flexibility for alternative focuses upon either tasks or data in designing the system. Vortex workflow indicates good understandability and corrigibility by defining declarative specifications and application layer language, so it is significantly important to popularize the workflow management system. In this paper, various lasic constituents and concepts in Vortex workflow are introduced some related issues are discussed, and finally an example about the modeling in Vortex is presented.

Keywords Schema Snapshot Enabling condition Side - effect Combining function Combining semantics language

1 引 言

传统的工作流管理系统为组织和协调多道任务的执行提供了控制机制,已有许多工作流模型以及它们的实现算法广泛地应用在商业与科学计算领域中。这类工作流着重于对任务的控制与管理,任务可以由软件系统或用户提交,工作流表示为直方图或各类 Petri 网系统。科学计算领域的最新实践表明,在说明规范上,工作流应同时起到数据流的作用,执行过程应注重对数据的采集与分析。

为适应这种新的需要,贝尔实验室的研究人员提出了 Vortex 工作流的概念。Vortex 工作流以属性为中心,工作流的整个执行过程都在于对属性的获取与定义。属性既可以模拟特定任务的执行状态,又可被视作某一任务的输出数据,这使得工作流设计者可以根据不同应用领域的特点,灵活地选择侧重于任务或数据的表示方式。

Vortex 工作流定义了一种说明式规范,可以使用应用层语言对工作流进行设计与建模。应用层语言非常简明,不同应用领域的非计算机专业人员也能方便地理解,甚至在必要时改动工作流。

2 Vortex 工作流

2.1 属性与模块

属性在 Vortex 工作流中起着中心作用,Vortex 工作流着重于对属性的获取与赋值。大部分属性都由工作流中的模块定义,每个模块可以定义一个或多个属性。模块可看作一个黑盒,实现一定的功能,而其内部的执行过程是对外界透明的。每个模块都有它的启动条件和输入属性,启动条件是布尔表达式,只有当启动条件为真时,对应的模块才可能执行。模块执行将输入属性作为参数,定义输出属性作为执行结果。工作流设计人员可以通过设置相应模块的启动条件,实现对多道任务的控制与协调。Vortex 工作流中并没有对控制流作显式定义,控制流可由数据流与模块启动条件间接推导出来。

Vortex 工作流中属性分为三种:源属性、目标属性和内部属性。源属性作为整个 Vortex 系统的输入,目标属性作为它的输出,属性集合中源属性与目标属性之外的属性称为内部属性。任何内部属性和目标属性

收稿日期:2001 - 03 - 17。杨俊,硕士生,主研领域:软件工程。

都必须由一个模块定义，即作为它的输出属性。Vortex 工作流中，模块 M 表示为 $M(A_1, A_2, \cdots, A_n; B_1, B_2, \cdots, B_k)$，其中 $A_1, A_2, \cdots, A_n$ 表示 M 的输入属性，$B_1, B_2, \cdots, B_k$ 表示 M 的输出属性，且要求 $\{A_1, A_2, \cdots, A_n\}$ 与 $\{B_1, B_2, \cdots, B_k\}$ 不相交。

模块的执行过程虽然对外界透明，但其输出可能对 Vortex 系统外部造成影响。特别是定义目标属性的模块，它的输出属性可能用于改变外界状态（如更新数据库状态等）。在 Vortex 工作流中，将输出属性用于更新系统外部状态的模块称为边际效应模块。

属性与模块具有不同的状态集合。

属性具有五种状态：

（1）UNINITIALIZED　无该属性的状态信息，表示定义该属性的模块也没有状态信息；

（2）VALUE　该属性具有有效值；

（3）EXC　定义该属性的模块执行发生异常，属性处于异常态；

（4）DISABLED　定义该属性的模块启动条件为 False，或模块执行后返回该态；

（5）FAIL　定义该属性的模块执行后未返回 VALUE/EXC/DISABLED 态时返回该态。

模块具有四种状态：

（1）UNINITIALIZED　模块启动条件无法确定或存在输入属性为 UNINITIALIZED 态；

（2）SUCCESS　模块成功执行结束；

（3）EXC　模块执行异常结束；

（4）DISABLED　模块启动条件为 False。

属性或模块处于非 UNINITIALIZED 态时称为稳定的，稳定态就是指属性或模块的状态不会再发生变迁。

2.2 Vortex 工作流模式

Vortex 工作流模式给出了 Vortex 工作流的形式化描述。为引出工作流模式，先定义它的预模式。

对于一个工作流，它的预模式是六元组 S = (Att, Src, Tgt, Mod, Cnd, Eff)。

（1）Att 是属性集合；

（2）Src, Tgt 分别表示 Vortex 系统的源属性与目标属性集合；

（3）Mod 是模块集合，满足：

a. $\forall M \in Mod, Input(M) \subset Att$

b. $\forall A \in (Att - Src), \exists M \in Mod$ 且只能有一个 M，令 $A \in Output(M)$

Input, Output 分别取 M 的输入和输出属性集合；

（4）Cnd 表示引用了 Att 中属性的模块启动条件集合；

（5）Eff 表示工作流中的边际效应模块集合，$Eff \subset Mod$。

每个确定的预模式，都有与它对应的有向依赖图 G。G 的结点集为 $Mod \cup Cnd$，边集为 $E_c \cup E_d$，E_c 为控制边集，E_d 为数据边集，定义如下：

$E_c = \{(M, \gamma_{M1}) \mid M, M_1 \in Mod$ 且 $\exists A \in Output(M)$ 被 γ_{M1} 引用$\} \cup \{(\gamma_{M1}, M_1) \mid M_1 \in Mod\}$

$E_d = \{(M, M_1) \mid M, M_1 \in Mod$ 且 $\exists A \in Output(M)$，满足 $A \in Input(M_1)\}$

γ_M 表示模块 M 的启动条件。

对于一个给定的工作流，如果预模式对应的有向依赖图没有回路，该预模式就称为 Vortex 模式，该工作流称为 Vortex 工作流。

显然，任何有向依赖图都应是连通的，Vortex 工作流中不应存在永远不可能被触发的模块和启动条件。依赖图的无回路性将在 2.3 中解释，这里先对预模式定义 3.b 作出解释：任一非源属性，必须由一个，也只能由一个模块定义。非源属性，即内部属性与目标属性必须由模块定义，是为了保证属性来源的完整性，它们都由工作流系统自身产生，不是外来的。同时，为了避免并发执行所造成的属性取值的不确定性，非源属性只能由一个模块定义。假定模块 M_1 与 M_2 都定义了属性 A，且工作流运行的某时刻两个模块同时被系统选中执行，那么显然 A 的最后取值是不确定的。

2.3 快　照

快照用于描述 Vortex 系统的运行状况，反映某一时刻工作流各个模块与属性的状态与取值，表示为三元组 $s = (\sigma, \mu, \xi)$。

（1）σ 为状态函数，定义域 $Mod \cup Att$ 上的全映射，满足：

a. $\forall M \in Mod, \sigma(M) \in \{$UNINITIALIZED, SUCCESS, EXC, DISABLED$\}$；

b. $\forall A \in Att, \sigma(A) \in \{$UNINITIALIZED, VALUE, EXC, DISABLED, FAIL$\}$；

（2）μ 为属性值函数，定义域 Att 上的部分映射，$\mu(A)$存在当且仅当 $\sigma(A) =$ VALUE；

（3）ξ 为异常值函数，定义域 $Mod \cup Att$ 上的部分映射，满足 $\forall X \in Mod \cup Att, \xi(X)$存在当且仅当 $\sigma(X) =$ EXC；

（4）σ 满足以下条件：

a. 若 $\sigma(M) \in \{$SUCCESS, EXC$\}$，则 M 定义的所有属性状态稳定；

b. 若 $\sigma(M) =$ SUCCESS，则由 M 定义的属性状态不可能是 FAIL；

c. 若 $\sigma(M) =$ DISABLED/UNINITIALIZED，则 M 定义的所有属性状态相应为 DISABLED/UNINITIALIZED。

Vortex 工作流中，有两类特殊的快照：初始快照与终止快照。初始快照下，所有源属性状态为 VALUE 或 DISABLED，所有模块与非源属性状态为 UNINITIAL-

IZED。终止快照下，所有模块与属性的状态稳定。直观地讲，Vortex 工作流可以看作一个虚拟机，对于特定的输入，将系统的初始快照映射到终止快照。可以证明，在 Vortex 工作流规范下，G 的无回路性是工作流执行始终结束于终止快照的充分条件。证明如下：

已知 G 无回路，假设工作流的最终状态下有一非源属性 A 状态为 UNINITIALIZED，即 $\sigma(A)$ = UNINITIALIZED，则定义 A 的模块 M_1 满足 $\sigma(M_1)$ = UNINITIALIZED，可知 $\exists B \in Att$ 被 γ_{M_1} 引用或作为 M_1 的输入属性，且 $\sigma(B)$ = UNINITIALIZED。因为 G 无回路，源属性在初始快照下为 VALUE 或 DISABLED 态，在执行过程中其状态与值均不再变化，所以 B 为非源属性，得到定义 B 的模块 M_2 且 $\sigma(M_2)$ = UNINITIALIZED。依此类推，由 G 的无回路性得到一个 UNINITIALIZED 态模块的无穷序列 $M_1, M_2, \cdots$，而这与 G 结点数有限，从而模块个数有限相矛盾，故假设不成立，工作流执行将始终结束于终止快照，证毕。

现在可以理解 2.1 中模块表示为 $M(A_1, A_2, \cdots, A_n; B_1, B_2, \cdots, B_k)$ 时，$\{A_1, A_2, \cdots, A_n\}$ 与 $\{B_1, B_2, \cdots, B_k\}$ 必不相交的原因，这也是为了避免 G 中回路的出现。G 的无回路性在工作流执行过程中，主要表现为非源属性一旦被某一模块定义，它的状态及取值(若存在的话)就不会再改变；同样，模块的启动条件一旦为 True，就不会再变为 False。

2.4 判定模块

判定模块是 Vortex 系统中的一类特殊模块，它的输出属性只有一个，模块与输出属性同名。判定模块包含一组属性规则及由联合语义语言(CSL)定义的联合函数。属性规则类似于高级程序语言中的条件语句，根据输入属性判定条件是否成立，成立则赋给属性一个临时值。与条件语句不同的是，对于给定的输入，可能会有多条属性规则符合，这样属性就取得多个临时值。联合函数则以临时值集合为参数生成属性的最终取值。联合函数体现的语义可以很复杂，对临时值作各种迭代计算；也可以按简单的规则(如取最大值，最小值等)取临时值中的一个。因此，联合函数的定义成为设置判定模块的关键。

联合语义语言作为一种语言，包含以下几种值类型。

(1) 原子类型：含整型、字符型与字符串型；

(2) 元组类型：$< a_1, a_2, \cdots, a_n >$，且 a_i 属于 CSL 值类型($1 \leqslant i \leqslant n$)；

(3) 列表类型：$[a_1, a_2, \cdots, a_n]$，且 a_i 为相同 CSL 值类型($1 \leqslant i \leqslant n$)；

(4) 包类型：$\{a_1, a_2, \cdots, a_n\}$，且 a_i 为相同 CSL 值类型($1 \leqslant i \leqslant n$)。

另外，空值表示为 $\perp$。Vortex 工作流要求所有属性的取值均为 CSL 值类型。

属性规则表示为 if cond then A←term，其中 cond 为条件判断，A 为属性名，term 为返回 CSL 值类型的表达式，term 可取常量，或属性集合中 A 以外的属性，或用户自定义函数。

CSL 定义了若干内部函数，可由用户定义的联合函数调用。

(1) proj[a_i]：作用于元组或列表，返回元素 a_i 的投影值；

(2) map[f]：作用于列表或包，f 为任意一元函数，map[f](X)表示由 f 作用于 X 中所有元素后返回的新的集合(列表或包)；

(3) collect[id,f]：作用于列表或包，f 为任意二元函数，collect[id,f](X)表示由 f 迭代作用于 X 中两个元素(X 为列表时，按序迭代)后返回的 CSL 值。若 X 为空，返回 id；若 X 中仅有一个元素，返回该元素的值；

(4) group[f,g]：作用于列表或包，f,g 为任意一元函数，group[f,g](X)返回元组的集合，元组形式为 $< v, S >$，v 为原子类型，S 为包，满足 $\forall < v, S > \in$ group[f,g](X)，$\exists x \in X$，有 $f(x) = v$ 且 $S = \{g(x) \mid f(x) = v\}$，故 group[f,g](X)的语义是按 f(x)的值对所有 g(x)进行分组。

2.5 操作语义

Vortex 工作流中，模块自 UNINITIALIZED 态被触发执行，在变迁为 SUCCESS 或 EXC 态之前，还有若干环节。现对模块状态集进行扩充，新增三个状态如下：

(1) READY：模块启动条件为 True 且输入属性稳定；

(2) LAUNCHED：模块被系统选中，已触发执行；

(3) EXECUTED：模块执行中止(成功或异常)，将要返回属性值。

模块状态集扩充后，状态变迁图如图 1 所示。

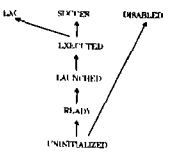

图 1

由图 1 可见，只有处于 READY 态的模块才可能被系统中执行。

模块状态的增加同时反映到快照上，此时的快照称为操作快照。Vortex 工作流的执行，可以定性为一个操作快照序列 $s_0, s_1, \cdots, s_n$，其中 s_0 为初始快照，s_n 为终止快照。值得注意的是，对于给定的初始快照 s_0，操作快照序列并非唯一。Vortex 系统定义了两个内部函数 Choice 及 inference。Choice 函数根据 Vortex 模式及当前快照选取处于 READY 态的模块；inference 函数并不触发模块，而是根据当前快照及系统信息(主要是当前已触发模块的信息)推导出下一快照。系统可由 Choice 及 inference 函数推导出整个操作快照序列，表述如下：

a. s_i = lauch(Choice(S,s_{i-1}),s_{i-1})或

b. s_i = inference(s_{i-1},info)。

lauch 函数是将快照 s_{i-1}下 Choice 返回模块的状态由 READY 变迁为 LAUNCHED,从而产生快照 s_i。

3 应用实例

在上海市高校收费系统的设计中,我们按 Vortex 工作流规范对系统进行分析与建模。

需求描述:根据学生学号系统反馈他的应收款记录,用户按该学生各项费用实付金额对应收款记录作调整,并计算该学生本次付款总额。系统根据该学生本次付款总额,他的银行可扣款额及可贷款额,推荐某种付款方式(含付款形式与金额)。假定现有付款形式有银行扣款,贷款和现金支付。用户按学生要求,采纳系统推荐的付款方式或另选其它付款方式生成该学生的实际到款项。同时,系统由当前可用发票列表,选取一个空白发票号。最后根据该学生的实际到款项及空白发票号,生成他的到款记录;根据该学生各项费用实付金额及空白发票号,生成他的发票记录。

经过细致分析后我们作出如下设计(详见表 1)

表 1

模块	模块描述	启动条件	输入属性	输出属性
M_1	读取应收款记录	True	Uni _ no	Ysk[].Fcode, Ysk[].Amt
M_2	调整付款金额	VALUE(Ysk[].Fcode) &VALUE(Ysk[].Amt)	Ysk[].Fcode, Ysk[].Amt	Ysk[].Yamt
M_3	计算付款总金额	True	Ysk[].Yamt	Total
M_4	推荐付款方式	Total>0	Total, Bank _ amt, Loan _ amt	Sug _ payment
M_5	获取空白发票号	Total>0	Ticket[]	Tno
M_6	选择付款方式并生成实际到款项	True	Sug _ payment, Total,Bank _ amt Loan _ amt	Payment
M_7	生成到款记录	True	Uni _ no,Tno, Payment,	Income
M_8	生成发票记录	True	Uni _ no,Tno, Ysk[].Fcode, Ysk[].Yamt	Ysk _ finished
M_9	返回'无收款项'错	EXC(Ysk[].Fcode) = 'DB _ DOWN' ‖ EXC(Ysk[].Amt) = 'DB _ DOWN'	Uni _ no	NoFeiItemError
M_{10}	返回'付款金额0'错	Total=0	Uni _ no	ZeroAmtError
M_{11}	获取银行可扣款额	True	Uni _ no	Bank _ amt
M_{12}	获取可贷款额	True	Uni _ no	Loan _ amt
M_{13}	获取可用发票列表	True	—	Ticket[]

由此产生 Vortex 工作流预模式 S = (Att,Src,Tgt,Mod,Cnd,Eff),其中 Att 为表 1 中所有属性的集合,Src 仅含 Uni _ no(学号),Tgt 含 Income(到款记录),Ysk _ finished(发票记录),NoFeiItemError('无收款项'错),ZeroAmtError('付款金额 0'错),Mod 含表 1 中 13 个模块,Cnd 含模块 M_2,M_4,M_5,M_9,M_{10}的启动条件,按序表示为 e_1,e_2,e_3,e_4,e_5,Eff 含模块 M_7 与 M_8,因为生成的到款记录与发票记录将改变外部数据库的状态。

与预模式 S 对应的有向依赖图 G 表示如图 2。

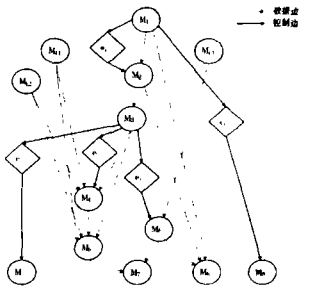

图 2

显然,图 2 满足无回路性,所以六元组 S 为 Vortex 模式。图 2 中模块 M_4 为判定模块,按应用层语言描述如下:

```
Module Sug _ payment{
Enabling Condition:Total > 0
Computation:Sug _ payment =
HPW({if Bank _ amt ≥ Total then Sug _ payment ← << bank,
      Total > ,3 > ;
      if Loan _ amt ≥ Total then Sug _ payment ← << loan,
      Total > ,2 > ;
      if Total > 0 then Sug _ payment ← << cash, Total > ,
      1 > })
}
```

其中形如 if 的语句表示属性规则,条件满足时生成属性 Sug _ payment 的临时值。临时值取元组形式 < s,p > ,s也为元组,表示付款形式及金额,p 表示权值。HPW 是自定义的联合函数,作用于 Sug _ payment 的临时值集合,语义为高权值命中,返回临时值集合中具最大 p 值项的元素的 s 项,定义为:

HPW(X) = proj[s](collect[⊥,psup](X))

psup(x,y) = if proj[p](x) ≥ proj[p](y) then x else y

X 为临时值集合,psup 为二元函数,返回 x,y 中 p 值项较大的元素。

整个判定模块的语义体现了各种付款方式受推荐的优先级,在银行可扣款额与可贷款额足够的情况下,银行扣款优先于贷款,贷款优先于现金支付。

这个实例中初始快照 s_0 下仅 Uni _ no 为 VALUE 态,正常的收费过程结束于终止快照 s_n,且 s_n 下 Ysk _ finished 与 Income 为 VALUE 态,NoFeiItemError 及 ZeroAmtError 为 DISABLED 态。Vortex 工作流的操作语义不是本文重点,对快照序列这里不作详细讨论。

从上例可以看到,Vortex 预模式和有向依赖图的定义简明,直观,易于理解。模块定义使用描述性的应用层语言,这有利于非计算机专业人员方便地理解,甚

(下转第 64 页)

(上接第 27 页)

至在必要的时候改动工作流。Vortex 工作流的易修改性表现在对于应用体的需求变化,工作流改动主要局限于对相关启动条件与判定模块的修正,而工作流模式在整体上保持相对稳定。以下我们根据收费系统的两种需求变动,对其 Vortex 工作流作相应调整。

变动 1 每个学生有一个缓交日期,系统仅对学生超过其缓交日期的费用项实施收款。我们只需新增源属性 Now(当前日期),M_1 新增输出属性 Suspend _ date(学生缓交日期),同时 e_1 改为 VALUE(Ysk[ ].Fcode)&VALUE(Ysk[ ].Amt)& Now > Suspend _ date,即可满足。

变动 2 付款形式限于银行扣款与现金支付,在银行可扣款额不足的情况下,系统仍然优先推荐银行扣款方式,但不足余额由现金支付。由此,我们对判定模块 M_4 改动如下:

```
Module Sug _ payment{
Enabling Condition: Total > 0
Computation: Sug _ payment =
HPW({if Bank _ amt ≥ Total then Sug _ payment ← << bank, Total > ,3 > ;
     if Bank _ amt > 0 and Bank _ amt < Total then Sug _ payment ← << bank, Bank _ amt > ,2 > ;
     if Bank _ amt > 0 and Bank _ amt < Total then
       Sug _ payment ← << cash, Total - Bank _ amt > ,2 > ;
     if Total > 0 then Sug _ payment ← << cash, Total > , 1 > })
}
```

令 temp = group[proj[p],proj[s]](X),X 为临时值集合,temp 中元素取元组形式 < v,w > ,

HPW(X) = proj[w](collect[⊥,psup](temp))

psup(x,y) = if proj[v](x) ≥ proj[v](y) then x else y

更新后的联合函数 HPW 语义为按权值对可能的付款方式进行分组,再取与最大权值项对应的付款方式集合。注意此时系统推荐的是一个付款方式集合,它可能包括多种付款形式,如银行扣款与现金支付。

4 结束语

Vortex 工作流以属性为中心的特性,能有效地适应工作流对任务与数据的不同侧重。说明式规范与应用层语言的引入,特别是判定模块中属性规则与联合函数的使用,使 Vortex 工作流有别于内嵌大量源代码的传统工作流,变得易于理解与修改,而这对工作流管理系统的推广应用将具有重大意义。本文运用 Vortex 工作流规范对学生收费系统进行了建模分析,以实践表明其在系统分析与设计过程中的使用价值。

Vortex 工作流完善的重点在于建立 Vortex 工作流引擎。引擎是一个应用程序生成器,同时作为用户自定义工作流的解释器,它提供从工作流描述到可运行系统的映射。良好的引擎还要能够支持本地系统中多道工作流的并发执行,或在分布式系统中协调不同应用体的工作流。以上这些,均有待进一步的研究。

参 考 文 献

[1] Hull R., Llirbat F., Simon E., Su J., Dong G., Kumar B. and Zhou G., Declarative Workflows that Support Easy Modification and Dynamic Browsing. Proc. Int. Joint Conf. on Work Activities Coordination and Collaboration, February, 1999.

[2] Fu X., Bultan T., Hull R. and Su J., Verification of Vortex Workflows. Proc. of Conf. on Tools and Algorithms for the Construction and Analysis of Systems(TACAS), February, 2001.

[3] Hull R., Llirbat F., Kumar B., Zhou G., Dong G. and Su J., Optimization Techniques for Data - intensive Decision Flows. IEEE Int. Conf. on Data Engineering(ICDE), San Diego, March, 2000.

[4] Cichocki A., Helal A., Rusinkiewicz M. and Woelk D., Workflow and Process Automation: Concepts and Technology. Boston, Kluwer Academic Publishers, 1998.

(上接第 46 页)

善。推理网络在知识库中都是以规则的形式来表现。

4 结束语

推理机的设计直接关系到医疗智能诊断专家系统的成败。本文主要讨论了初步诊断和鉴别诊断推理机的设计和实现过程。推理采用了正、反向推理相结合的方式,其优点是:提高了推理效率、推理方法类似于医学专家的诊断思维过程,能满足关于推理效果的四点要求。在初步诊断推理过程中,采用不精确推理模式来处理证据的模糊性知识的不确定性因果关系,并用专家评分"加权求和"法来表示这种模糊性或不确定性,使推理更接近于专家思维。

该推理机的设计方法已在医学专家系统——遗传及先天性疾病智能诊断系统中获得了成功的应用,并取得了令人满意的效果,在临床试验中,成功率达 90% 以上。

参 考 文 献

[1] 魏世成、王翰章,"医学专家思维的计算机模拟——诊断类医学专家系统的设计与实现",《计算机应用》,No.4,1990."

[2] 施鸿宝、王秋荷,专家系统,西安交通大学出版社,1990.

[3] V. Wuwngse, "Experiences in the Development of A Medical Expert System", Proc. of MEDINFO 89, NorthHolland, 1989.

[4] Ameshs. Patil, "Causal Reasoning in Computer Programs for Medical Diagnosis", Computer Methods and Programs in Biomedicine, 1987.