知识库维护过程中检查其协调性的有效方法*

沈宁川 龙 翔 李 未

(北京航空航天大学计算机系 北京 100083)

摘要 本文首先描述了知识库维护过程中的协调性问题,然后给出了扩充逻辑程序设计的框架,在此框架下,每个逻辑程序等价于一个知识库.为了检查知识库的协调性,本文为知识库中的推理规则构造了正支持集和负支持集,并给出了一些定义;基于这些概念和定义,提出了知识库维护过程中检查知识库协调性的一种有效方法,并证明了相关的定理;基于此方法,实现了一个算法CHIME,并给出了用CHIME分析一些知识库的实验结果.本文还提到一些相关的工作,最后给出结论.

关键词 知识库,协调性检查,扩充逻辑程序设计,实现.

可靠性(Reliability)是维护知识库系统过程中的重要问题.这是一个通常称为证实(Validation)的问题.知识库系统证实的重要部分是确保知识库的协调性,即从知识库中推不出矛盾的结果.

在知识库的维护过程中,尽管知识库在某一阶段是协调的,但加入一个新的事实到知识库中、或从知识库中删除一个已有的事实会导致知识库产生矛盾.这意味着知识库维护的结果影响了知识库的结论.

上述问题可以用以下方式描述.假设$\Gamma_0,\Gamma_1,\cdots,\Gamma_n,\cdots$是一个知识库序列,且$\Gamma_n$是协调的(这里$\Gamma_n$是扩充逻辑程序).我们的问题是:(1) 是否$\Gamma_n \nvdash \neg A$蕴含$\Gamma_{n+1}$是协调的,这里$\Gamma_{n+1}=\Gamma_n \cup \{A\}$.(2) 是否$A \in \Gamma_n$蕴含$\Gamma_{n+1}$是协调的,这里$\Gamma_{n+1}=\Gamma_n-\{A\}$.让我们考察下面例子.

例1:令$\Gamma_1=\{A \leftarrow B, D, \neg A \leftarrow B, D\}$是一知识库.明显地,$\Gamma_1 \nvdash \neg B$且$\Gamma_1$是协调的.令$\Gamma_2=\Gamma_1 \cup \{B\}$,因为$\Gamma_2 \vdash A$且$\Gamma_2 \vdash \neg A$,所以$\Gamma_2$不协调.

例2:令$\Gamma_1=\{A \leftarrow notC, \neg A \leftarrow B, B, C\}$是一知识库.这里连接词*not*的含义是否定即失败(Negation-as-failure).明显地,$C \in \Gamma$且Γ_1是协调的.令$\Gamma_2=\Gamma_1-\{C\}$,则Γ_2是不协调的,因为$\Gamma_2 \vdash A$且$\Gamma_2 \vdash \neg A$.

---

* 本文研究得到国家863高科技项目基金资助.
作者**沈宁川**,1964年生,博士,讲师,主要研究领域为人工智能,软件工程,计算机通信.**龙翔**,1963年生,博士,副教授,主要研究领域为人工智能,计算机体系结构,VLSI设计.**李未**,1943年生,博士,教授,博士导师,主要研究领域为计算机科学理论,人工智能基础.

本文通讯联系人:沈宁川,北京100083,北京航空航天大学计算机系

本文1995-12-22收到修改稿

在上述例子中,明显地,维护的结果影响了知识库的结论.

本文提出了在知识库维护过程中检查知识库协调性的一种有效方法,并基于这种方法实现了一个算法,而且给出了用该算法分析一些知识库的实验结果.

本文第 1 节给出了扩充逻辑程序设计的框架. 第 2 节为知识库中的推理规则构造了正支持集 PSS(positive support set)和负支持集 NSS(negative support set),并给出了一些定义;基于这些概念和定义,提出了知识库维护过程中检查知识库协调性的有效方法,并证明了相关的定理. 第 3 节基于上述方法,实现了一个算法 CHIME,并给出了用 CHIME 分析一些知识库的实验结果. 第 4 节提到一些相关的工作,最后给出结论.

1　扩充逻辑程序设计的框架

传统逻辑程序设计可以作为表达知识的工具,但它的一个很大局限是不允许直接处理不完全信息. 为了克服这一局限,人们提出了扩充逻辑程序设计.[1]在扩充逻辑程序设计中,除了包含**否定即失败**(Negation-as-failure) (*not*)外,还包含**经典否定**(Classic Not) ($\neg$).

根据 Herbrand 域上所有不带变量的实例的集合,我们可以给出包含变量的扩充逻辑程序设计的语义. HB 表示 *Herbrand* 基. HB_P 表示集合 $HB\cup\{\neg A\mid A\in HB\}$,$HB_n$ 表示集合 $\{not\ A\mid A\in HB_P\}$,Lit 表示 $HB_P\cup HB_n$. 这里只考虑非循环程序.

扩充逻辑程序设计的框架是一偶对 $\langle L,R\rangle$,其中

- $L=\{\bot\}\cup Lit$
  $\cup\{A\leftarrow B_1,\cdots,B_n\mid A\in HB_p,B_1,\cdots,B_n\in Lit,\text{且 } n\geqslant 0\}$
  $\cup\{A\leftarrow B_1;\cdots;B_n\mid A\in HB_p,B_1,\cdots,B_n\in Lit,\text{且 } n\geqslant 0\}$,
- R 是下列形式的推理规则的集合

$$\frac{A\leftarrow B_1,\cdots,B_n\quad B_1,\cdots,B_n}{A}$$

$$\frac{A\leftarrow B_1;\cdots;B_n\quad B_1;\cdots;B_n}{A}$$

$$\frac{A\quad notA}{\bot}$$

这里 $A\in HB_p,B_1,\cdots,B_n\in Lit$,且 $n\geqslant 0$.

当 $n=0$,A 是一事实. 规则 $A\leftarrow B_1,\cdots,B_n$ 的语义解释是 $B_1\wedge\cdots\wedge B_n\supset A$;规则 $A\leftarrow B_1;\cdots;B_n$ 的语义解释是 $B_1\vee\cdots\vee B_n\supset A$.

扩充逻辑程序 P 是一理论,其中 $P\subseteq L$. 在这一框架下,P 等价于一个知识库 Γ.

定义 1.1(**规则的头**). 令 Γ 是一知识库. A 称为规则 $A\leftarrow B_1,\cdots,B_n$ 或 $A\leftarrow B_1;\cdots;B_n$ 的头. $H(\Gamma)$记为 Γ 中所有规则的头的集合.

2　协调性检查

在本节中,知识库 Γ 等价于一个扩充逻辑程序.

定义 2.1(**正支持集和负支持集**). 令 Γ 是一知识库. 如果 $A\in H(\Gamma)$,A 的正支持集和

负支持集分别表示为 $PSS^A(\Gamma)$和 $NSS^A(\Gamma)$.

为构造集合 $PSS^A(\Gamma)$,函数 $Plength(X,\Gamma)$,$Pla(X,\Gamma)$及 $Plo(X,\Gamma)$递归定义如下:

(i) $Plength(A,\Gamma)=0$,如果 $A\in\Gamma$ 或 $A\notin H(\Gamma)$.

(ii) $Plength(A,\Gamma)=Pla(B_1,\Gamma)\times\cdots\times Pla(B_n,\Gamma)$,如果 $A\leftarrow B_1,\cdots,B_n\in\Gamma$.

(a) $Pla(B_i,\Gamma)=1$,如果 $B_i\in HB_n$ 或 $Plength(B_i,\Gamma)=0$.

(b) $Pla(B_i,\Gamma)=Plength(B_i,\Gamma)$,其它情况.

(iii) $Plength(A,\Gamma)=Plo(B_1,\Gamma)+\cdots+Plo(B_n,\Gamma)$,如果 $A\leftarrow B_1;\cdots;B_n\in\Gamma$.

(a) $Plo(B_i,\Gamma)=0$,如果 $B_i\in HB_n$.

(b) $Plo(B_i,\Gamma)=Plength(B_i,\Gamma)$,如果 $B_i\in HB_p$.

这里 $1\leqslant i\leqslant n$.

同样,为构造集合 $NSS^A(\Gamma)$,函数 $Nlength(X,\Gamma)$,$Nla(X,\Gamma)$及 $Nlo(X,\Gamma)$递归定义如下:

(i) $Nlength(A,\Gamma)=0$,如果 $A\in\Gamma$ 或 $A\notin H(\Gamma)$.

(ii) $Nlength(A,\Gamma)=Nla(B_1,\Gamma)\times\cdots\times Nla(B_n,\Gamma)$,如果 $A\leftarrow B_1,\cdots,B_n\in\Gamma$.

(a) $Nla(B_i,\Gamma)=1$,如果 $B_i\in HB_n$ 或 $Nlength(B_i,\Gamma)=0$.

(b) $Nla(B_i,\Gamma)=Nlength(B_i,\Gamma)$,其它情况.

(iii) $Nlength(A,\Gamma)=Nlo(B_1,\Gamma)+\cdots+Nlo(B_n,\Gamma)$,如果 $A\leftarrow B_1;\cdots;B_n\in\Gamma$.

(a) $Nlo(B_i,\Gamma)=1$,如果 $B_i\in HB_n$.

(b) $Nlo(B_i,\Gamma)=Nlength(B_i,\Gamma)$,如果 $B_i\in HB_P$.

这里 $1\leqslant i\leqslant n$.

集合 $PSS^A(\Gamma)$可以用下面的方式递归构造:

(i) $PSS^A(\Gamma)=\{\}$,如果 $A\in\Gamma$ 或 $A\notin H(\Gamma)$.

(ii) $PSS^A(\Gamma)=\{PSS_1^A(\Gamma),\cdots,PSS_u^A(\Gamma)\}$,如果 $A\leftarrow B_1,\cdots,B_n\in\Gamma$.

这里 $u=Plength(A,\Gamma)$,$PSS_t^A(\Gamma)=\bigcup_{i=1}^{n}PA_i(1\leqslant t\leqslant u)$.

(a) $PA_i=PSS_{t_i}^{B_i}(\Gamma)\cup\{B_i\}$,如果 $B_i\in HB_P$ 并且 $Plength(B_i,\Gamma)\neq 0$.

$(1\leqslant t_i\leqslant Plength(B_i,\Gamma))$

(b) $PA_i=\{B_i\}$,如果 $B_i\in HB_P$ 并且 $Plength(B_i)=0$.

(c) $PA_i=\varnothing$, $B_i\in HB_n$.

(iii) $PSS^A(\Gamma)=\bigcup_{i=1}^{n}PO_i$,如果 $A\leftarrow B_1;\cdots;B_n\in\Gamma$.

(a) $PO_i=PSS^{B_i}(\Gamma)\cup\{B_i\}$,如果 $B_i\in HB_p$.

(b) $PO_i=\varnothing$,如果 $B_i\in HB_n$.

这里 $1\leqslant i\leqslant n$.

集合 $NSS^A(\Gamma)$可以用下面方式递归构造:

(i) $NSS^A(\Gamma)=\{\}$,如果 $A\in\Gamma$ 或 $A\notin H(\Gamma)$.

(ii) $NSS^A(\Gamma)=\{NSS_1^A(\Gamma),\cdots,NSS_v^A(\Gamma)\}$,如果 $A\leftarrow B_1,\cdots,B_n\in\Gamma$.

这里 $v=Nlength(A,\Gamma)$,$NSS_t^A(\Gamma)=\bigcup_{i=1}^{n}NA_i(1\leqslant t\leqslant v)$.

(a) $NA_i=NSS^{B_i}_{t_i}(\Gamma)$,如果 $B_i\in HB_p$. $(1\leqslant t_i\leqslant Nlength(B_i,\Gamma))$.

(b) $NA_i=\{B_i'\}$,如果 $B_i\in HB_n$ 并且 $B_i=not\ B_i'$.

(iii) $NSS^A(\Gamma)=\bigcup_{i=1}^{n}NO_i$,如果 $A\leftarrow B_1;\cdots;B_n\in\Gamma$.

(a) $NO_i=NSS^{B_i}(\Gamma)$,如果 $B_i\in HB_p$.

(b) $NO_i=\{B_i'\}$,如果 $B_i\in HB_n$ 并且 $B_i=not B_i^1$.

这里 $1\leqslant i\leqslant n$.

例 2.1:令 $\Gamma=\{A\leftarrow A_1;A_2,\ A_1\leftarrow B_1,B_2,\ A_2\leftarrow B_3,\ B_1\leftarrow C_1,\neg C_2,not\ D,\ B_2\leftarrow C_3,C_4,not\ E,\ B_3\leftarrow\neg C_5,not\ F,\ \neg A\leftarrow D,E,F,not\ B_1,not\ \neg C_5\}$是一知识库,

我们有 $PSS^A(\Gamma)=\{\{A_1,B_1,B_2,C_1,\neg C_2,C_3,C_4\}_1^A,\{A_2,B_3,\neg C_5\}_2^A\}$,

$NSS^A(\Gamma)=\{\{D,E\}_1^A,\{F\}_2^A\}$,

$PSS^{\neg A}(\Gamma)=\{\{D,E,F\}_1^{\neg A}\}$,

$NSS^{\neg A}(\Gamma)=\{\{B_1,\neg C_5\}_1^{\neg A}\}$.

性质 2.1. 在知识库 Γ 中,令 $A\in H(\Gamma)$,

(*N-Cond*)　存在 $P\in PSS^A(\Gamma)$ 且对任意的 $\alpha\in p,\Gamma\vdash\alpha$. 则(*N-Cond*)是 $\Gamma\vdash A$ 的必要条件.

证明:如果 $\Gamma\vdash A$ 成立,则存在一个证明,在这个证明中,所用到的事实都是 Γ 中的事实,所用到的规则都是 Γ 中的规则. 如果把这个证明过程中的所有用到的属于 HB_p 文字作为一个集合 P,则 P 正是我们所定义的 A 的正支持集中的一个元素,即 $P\in PSS^A(\Gamma)$(参见定义 2.1). 因为 P 是依据一个证明所构造的,所以对每个 $\alpha\in P,\alpha$ 是可证的,即 $\Gamma\vdash\alpha$ 成立,因而条件(*N-Cond*)成立. 即(*N-Cond*)是 $\Gamma\vdash A$ 的必要条件. 以下用一个例子说明这个证明过程.

假设 $\Gamma=\{A\leftarrow B_1,B_2,\ B_1,\ B_2\}$. 根据 *PSS* 的构造方法,我们有 $PSS^A(\Gamma)=\{\{B_1,B_2\}\}$. 令 $\alpha\in\{B_1,B_2\}$,因为 $\Gamma\vdash A$,所以 $\Gamma\vdash\alpha$. 因此,(*N-Cond*)是 $\Gamma\vdash A$ 的必要条件. □

性质 2.2. 在知识库 Γ 中,令 $A\in H(\Gamma)$,

(*E-Cond*)对任意的 $N\in NSS^A(\Gamma)$,都存在一个 β 使得 $\beta\in N,\Gamma\vdash\beta$,则(*E-Cond*)是 $\Gamma\nvdash A$ 的充分条件.

证明:类似性质 2.1 的证明过程. 这里只用一个例子来说明这个过程.

假设 $\Gamma=\{A\leftarrow A_1;not A_2,A_1\leftarrow notB,\ notC,\ A_2\leftarrow notB,notD,\ B\}$. 根据 *NSS* 的构造方式,我们有 $NSS^A(\Gamma)=\{\{B,C\},\{B,D\}\}$. 显然存在一个 β,这里 $\beta=B$,使得 $\beta\in\{B,C\},\beta\in\{B,D\}$. 因为 $\Gamma\vdash\beta$ 所以 $\Gamma\nvdash A$. 因此,(*E-Cond*)是 $\Gamma\nvdash A$ 的充分条件. □

定义 2.2(**互补规则**). 令 Γ 是一知识库. 如果 $A\in H(\Gamma)$ 且 $\neg A\in H(\Gamma)$,称 A 和 $\neg A$ 对应的规则为互补规则(Complementary Rule),称 A 和 $\neg A$ 为互补规则的头. $CH(\Gamma)$ 用来表示 Γ 中所有互补规则的头的集合.

如果 $A\in CH(\Gamma)$,则 $\neg A\in CH(\Gamma)$. 在这种情况下,假设

$PSS^A(\Gamma)=\{PSS_1^A(\Gamma),\cdots,PSS_m^A(\Gamma)\}$ $(m=Plength(A,\Gamma))$,

$NSS^A(\Gamma)=\{NSS_1^A(\Gamma),\cdots,NSS_n^A(\Gamma)\}$ $(n=Nlength(A,\Gamma))$,

$PSS^{\neg A}(\Gamma)=\{PSS_1^{\neg A}(\Gamma),\cdots,PSS_u^{\neg A}(\Gamma)\}$ $(u=Plength(\neg A,\Gamma))$,

$NSS^{\neg A}(\Gamma)=\{NSS_1^{\neg A}(\Gamma),\cdots,NSS_v^{\neg A}(\Gamma)\}$ $(v=Nlength(\neg A,\Gamma))$.

定义 2.3(**良行为的互补规则**). 令 Γ 是一知识库,且假定 $A\in CH(\Gamma)$. 对任意 $PSS_i^A(\Gamma)$及 $NSS_j^{\neg A}(\Gamma)$,这里 $1\leqslant i\leqslant m$ 且 $1\leqslant j\leqslant n$,如果存在一个元素 α 使得

$$\alpha\in PSS_i^A(\Gamma)\cup\{A\}\text{且}\alpha\in NSS_j^{\neg A}(\Gamma),$$

或对任意 $PSS_i^{\neg A}(\Gamma)$及 $NSS_j^A(\Gamma)$,这里 $1\leqslant i\leqslant u$ 且 $1\leqslant j\leqslant n$,如果存在一个元素 α 使得

$$\alpha\in PSS_i^{\neg A}(\Gamma)\cup\{\neg A\}\text{且}\alpha\in NSS_j^A(\Gamma),$$

则称 A 和 $\neg A$ 对应的规则为良行为的互补规则(Well-behaved Complementary Rule),称 A 和 $\neg A$ 为良行为的互补规则的头.

明显地,在例 2.1 中,A 和 $\neg A$ 是良行为的互补规则的头.

定义 2.4(**攻击的互补规则**). 令 Γ 是一知识库,且假定 $A\in CH(\Gamma)$. 如果 A 和 $\neg A$ 不是良行为的互补规则的头,且存在 $PSS_i^A(\Gamma)$和 $PSS_j^{\neg A}(\Gamma)$(这里 $1\leqslant i\leqslant m$ 且 $1\leqslant j\leqslant n$),使得

$$PSS_i^A(\Gamma)\subseteq PSS_j^A(\Gamma)\text{或}PSS_j^A(\Gamma)\subseteq PSS_i^A(\Gamma)\text{成立},$$

则称 A 和 $\neg A$ 对应的规则为攻击的互补规则(Attacked Complementary Rule),称 A 和 $\neg A$ 为攻击的互补规则的头.

在例 1 和例 2 中,A 和 $\neg A$ 是攻击的互补规则的头.

定理 2.1. 在知识库 Γ 中,对任意 $A\in CH(\Gamma)$,如果 A 和 $\neg A$ 是良行为的互补规则的头,则 Γ 是协调的.

证明:假设 Γ 是不协调的,即 $\Gamma\vdash A$ 且 $\Gamma\vdash\neg A$.

根据定义 2.3,

(i) 如果对任意 $PSS_i^A(\Gamma)$及 $NSS_j^{\neg A}(\Gamma)$,这里 $1\leqslant i\leqslant m$ 且 $1\leqslant j\leqslant n$,存在一个元素 α 使得 $\alpha\in PSS_i^A(\Gamma)\cup\{A\}\text{且}\alpha\in NSS_j^{\neg A}(\Gamma)$,

因为 $\Gamma\vdash A$,所以 $\Gamma\vdash\alpha$(根据性质 2.1),因此 $\Gamma\nvdash\neg A$(根据性质 2.2). 与假设相矛盾.

(ii)或对任意 $PSS_i^{\neg A}(\Gamma)$且 $NSS_j^A(\Gamma)$,这里 $1\leqslant i\leqslant u$ 且 $1\leqslant j\leqslant n$,如果存在一个元素 α 使得 $\alpha\in PSS_j^{\neg A}(\Gamma)\cup\{\neg A\}\text{且}\alpha\in NSS_j^A(\Gamma)$,

因为 $\Gamma\vdash\neg A$,所以 $\Gamma\vdash\alpha$(根据性质 2.1),因此 $\Gamma\nvdash A$(根据性质 2.2). 与假设相矛盾.

所以,Γ 是协调的. □

定理 2.2. 在知识库 Γ 中,对任意的 $A\in CH(\Gamma)$,如果 A 和 $\neg A$ 是良行为的互补规则的头,$B\notin H(\Gamma)$且 $\neg B\notin H(\Gamma)$,则 $\Gamma\cup\{B\}$是协调的.

证明:因为 $B\notin CH(\Gamma)$,所以 $CH(\Gamma\cup\{B\})=CH(\Gamma)$. 因此,对任意 $A\in CH(\Gamma)$并且 $\neg A\in CH(\Gamma)$,我们有 $A\in CH(\Gamma\cup\{B\})$ 且 $\neg A\in CH(\Gamma\cup\{B\})$,根据 PSS 和 NSS 的构造方法,因为 $B\notin\Gamma$,所以 $Plength(B,\Gamma)=Plength(B,\Gamma\cup\{B\})=0$ 且 $Nlength(B,\Gamma)=Nlength(B,\Gamma\cup\{B\})=0$. 因此我们有

$$PSS^A(\Gamma\cup\{B\})=PSS^A(\Gamma)\text{且}NSS^A(\Gamma\cup\{B\})=NSS^A(\Gamma),$$

$$PSS^{\neg A}(\Gamma\cup\{B\})=PSS^{\neg A}(\Gamma)\text{且}NSS^{\neg A}(\Gamma\cup\{B\})=NSS^{\neg A}(\Gamma),$$

由于 Γ 是协调的,所以 $\Gamma\cup\{B\}$是协调的. □

定理 2.3. 在知识库 Γ 中,对任意 $A\in CH(\Gamma)$,如果 A 和 $\neg A$ 良行为的互补规则的头且 $B\in\Gamma$,则 $\Gamma-\{B\}$ 是协调的.

证明:

(i)如果 $\neg B\notin CH(\Gamma)$,那么 $CH(\Gamma-\{B\})=CH(\Gamma)$. 所以对任意 $A\in CH(\Gamma)$ 且 $\neg A\in CH(\Gamma)$,我们有 $A\in CH(\Gamma-\{B\})$ 且 $\neg A\in CH(\Gamma-\{B\})$,根据 PSS 和 NSS 的构造方法,因为 $B\notin\Gamma-\{B\}$,所以 $Plength(B,\Gamma)=Plength(B,\Gamma-\{B\})=0$ 且 $Nlength(B,\Gamma)=Nlength(B,\Gamma-\{B\})=0$. 因此,我们有

$$PSS^{A}(\Gamma-\{B\})=PSS^{A}(\Gamma)\text{ 且 }NSS^{A}(\Gamma-\{B\})=NSS^{A}(\Gamma),$$

$$PSS^{\neg A}(\Gamma-\{B\})=PSS^{\neg A}(\Gamma)\text{ 且 }NSS^{\neg A}(\Gamma-\{B\})=NSS^{\neg A}(\Gamma),$$

(ii) 如果 $\neg B\in CH(\Gamma)$,那么 $CH(\Gamma-\{B\})=CH(\Gamma)-\{\neg B, B\}$. 所以对任意 $A\in CH(\Gamma)$ 且 $\neg A\in CH(\Gamma)$,如果 $A\neq B$,那么我们有 $A\in CH(\Gamma-\{B\})$ 且 $\neg A\in CH(\Gamma-\{B\})$,类似(i),可得

$$PSS^{A}(\Gamma-\{B\})=PSS^{A}(\Gamma)\text{ 且 }NSS^{A}(\Gamma-\{B\})=NSS^{A}(\Gamma),$$

$$PSS^{\neg A}(\Gamma-\{B\})=PSS^{\neg A}(\Gamma)\text{ 且 }NSS^{\neg A}(\Gamma-\{B\})=NSS^{\neg A}(\Gamma),$$

又因为 Γ 是协调的,所以 $\Gamma-\{B\}$ 是协调的. □

3 实验结果

基于上述方法,我们用 C 实现了一个算法 CHIME,并用 CHIME 分析了一些由计算机随机产生的知识库.

表 1　用 CHIME 分析知识库的实验结果

互补规则	命题	循环规则	时间(ms)
200	1 275	24	66
400	2 581	0	149
600	3 845	42	249
800	5 043	2	516
1 200	7 440	32	766

表 1 列出了在 Sun Sparc2 工作站上用 CHIME 分析这些知识库的实验结果. 第 1 列是互补规则的数目;第 2 列是命题的数目;第 3 列是循环规则的数目;第 4 列是用 CHIME 分析这些知识库的执行时间.

4 相关的工作

关于知识库质量研究的重要部分是知识库协调性的概念. 这些工作给出了知识库必须满足的标准以便考虑验证. Suwa 等人的研究[2](以后有 Nguyen 等人扩充[3])是这一领域的第 1 个结果. 他们提出了基于规则两两比较的方法. Ginsberg[4]和 Rousset[5]独立提出了一个考虑所有规则库的演绎链的独到方法. 他们都用到基于 ATMS[6]的技术. Nonfjall 和 Larsen[7]提出了一个计算方法,该方法用到了逆绎推理中的反向链. Meseguer[8]给出了另一种基于 Petri 网的方法.

5 结 论

本文提出了检查知识库协调性的一种有效方法，并基于该方法实现了一个算法. 由于一般的关于知识库协调性检查算法效率不高，所以它们不适用于知识库系统的综合编辑器. 综合编辑器不同于一般的编辑器. 一般的编辑器只是一个用来编辑文本文件的工具，而它并不关心文件的内容. 在知识库系统中，综合编辑器不仅编辑一个正文文件(这里，该文本文件是一知识库)，而且还要检查文件内容的属性，如知识库中规则语法的正确性及知识库的协调性等等. 本文的实验结果表明我们提出的算法效率较高，该算法可以用在知识库系统的综合编辑器中. 如何将这一算法用在知识库系统的综合编辑器中已超出了本文的范围，详细内容见文献[9].

致谢 作者对张玉平博士的帮助表示感谢.

参考文献

1 Gelfond M, Lifschitz V. Logic programs with classical negation. In: Proc. 7th International Conference on Logic Programming, 1990. 579～597.

2 Suwa M, Scott A, Shortliffe H. An approach to verifying completeness and consistency in a rule-base expert system. Artificial Intelligence Magazine, 1982. 16～21.

3 Nguyen T A, Perkins W A, Laffey T J *et al*. Checking an expert system knowledge base for consistency and completeness. In: Proc. of the International Joint Conference on Artificial Intelligence, 1985. 375～379.

4 Ginsberg A. Knowledge base reduction: a new approach to checking knowledge base for inconsistency and redundancy. In: Proc. of the American Association of Artificial Intelligence, 1988. 585～589.

5 Rousset M C. On the consistency of knowledge base: the COVADIS system. In: Proc. of the European Conference on Artificial Intelligence, 1988. 79～84.

6 Kleer J de. An assumption-based TMS. Artificial Intelligence, 1986, **28**: 217～230.

7 Nonfjall H, Larsen H L. Detection of potential inconsistencies in knowledge bases. International Journal of Intelligent system, 1992, **7**: 81～96.

8 Meseguer P. A new method to checking rule bases for inconsistency: a petri net approach. In: Proc. of the European Conference on Artificial Intelligence, 1990. 437～442.

9 沈宁川. 基于开放逻辑的知识库维护[博士论文]. 北京航空航天大学, 1995.

AN EFFECTIVE METHOD OF CHECKING KNOWLEDGE BASES FOR CONSISTENCY IN THEIR MAINTENANCE

SHEN Ningchuan LONG Xiang LI Wei

(*Department of Computer Science Beijing University of Aero. & Astro. Beijing* 100083)

Abstract In this paper, the problem of the consistency of knowledge bases in their maintenance is first described, and then a framework for extended logic programming is

described, where an extended logic program is equivalent to a knowledge base. In order to check the consistency of a knowledge base, the PSS(positive support set) and the NSS (negative support set) for an inference rule in the base are constructed, and some definitions are given. Based on these concepts and definitions, an effective method of checking knowledge bases for the consistency in their maintenance is presented, and the related theorems are proved. Based on this method, an algorithm, called CHIME, is implemented, and the experimental results for analyzing some knowledge bases in CHIME are shown. Some related work is also mentioned. Finally, the conclusion of this paper is given.

Key words Knowledge base, consistency checking, extended logic programming, implementation.