知识标记与自动推理 Web 服务的研究*)

王家兵[1] 张 军[2] 彭 宏[1]
(华南理工大学计算机科学与工程学院 广州 510641)[1]
(广东商学院信息学院 广州 510320)[2]

摘 要 Internet 环境下的知识表示与自动推理是广大人工智能及相关领域研究者面临的新课题。为了给自动推理领域提供一个统一的问题(知识)表示格式及为 Web 用户提供自动推理 Web 服务,提出了一个基于多种类逻辑的知识标记语言 MSKML(Many Sorted Knowledge Markup Language 的简写),实现了使用 RLD(Rightmost Linear Deduction)演绎策略的多种类谓词演算的 COM 对象 Prover 来提供自动推理 Web 服务 ProverService。ProverService 采用简单对象访问协议 SOAP(Simple Object Access Protocol)作为服务器与 Web 客户间的网络通信协议,用 Web 服务描述语言 WSDL(Web Service Description Language)对其提供的服务进行了描述。最后,运用 RLD 演绎成功求解了著名的人工智能问题—Steamroller 问题,并以它为例,描述了该问题及 RLD 反驳的 MSKML 表示以及自动推理 Web 服务的调用过程。

关键词 知识表示,分布式自动推理,Web 服务,可扩展标记语言,简单对象访问协议

Research on Knowledge Markup and Automated Reasoning Web Service

WANG Jia-Bing[1] ZHANG Jun[2] PENG Hong[1]
(School of Computer Science and Engineering, South China University of Technology, Guangzhou 510641)[1]
(School of Information Science, Guangdong Commerce College, Guangzhou 510320)[2]

Abstract The knowledge representation and automated reasoning on the Web are an important issue for artificial intelligence community. In order to provide automated reasoning community with a uniform format of knowledge representation, a many-sorted knowledge markup language (abbr. MSKML) is proposed. Furthermore, in order to provide Web users with automated reasoning Web service, a prototype system of many-sorted predicate calculus with RLD (Rightmost Linear Deduction), i. e., the COM object Prover, is implemented. The Prover has a Web service ProverService that uses the SOAP (Simple Object Access Protocol) as the communication protocol between the Prover and Web users. The information of ProverService is described using WSDL (Web Service Description Language). When a client on the Web need the automated reasoning services, it represents the problem as a XML (Extensible Markup language) file conforming to the MSKML DTD (Document Type Definition) and sends a SOAP request message including the XML file to the automated reasoning server, and the automated reasoning server represents the solution as a XML file conforming to the MSKML DTD and send it back to the client as the SOAP response message. Finally, the well-known artificial intelligence problem—the Steamroller problem—is successfully solved by Prover and the MSKML representations of the problem and RLD refutation are given to illustrate our work.

Keywords Knowledge representation, Distributed automated reasoning, Web service, XML, SOAP

1 引言

Internet 的快速发展给人工智能提供了前所未有的发展机遇,同时也提出了新的挑战。其中,Internet 环境下的知识表示与自动推理是人工智能研究者面临的新课题。无论是面向大众的各种 Web 应用,还是面向专业界的分布式人工智能、分布式计算及分布式处理等等,知识(或数据)的表示都是其基本问题之一。本文从自动推理的角度,来探讨 Internet 环境下的知识表示与 Web 服务问题。

自动推理系统的性能测试与评价是自动推理领域备受关注的问题[1]。为了提出更为有效的自动推理原理以及开发新的推理系统,必须对诸如现有系统适应的问题领域,为什么在同等资源(内存、时间)限制下,有的系统能够解决别的系统所不能解决的问题等问题进行深入的分析。由于理论分析非常困难,因此在多数情形下,进行系统的实验分析就成为唯一选择。为进行公正的系统评价,需要一系列的评价标准以便系统测试者作出合理、准确、公正的评价结论。其中,问题的表示是其中主要原则之一[1],也就是说,系统的性能不能依赖于特定的问题表示格式。然而现有的系统具有各自的问题表示格式,当把同一个问题提交给不同的系统时,需要进行问题表示格式的转换。因此,提供一个统一的问题表示格式对系统评价是非常必要的。

对许多 Web 用户而言,World Wide Web 已成为获取各种信息与资源的"金矿"与"急救所"—他们总试图通过敲入

*)本工作得到广东省自然科学基金(编号 020199)、华南理工大学自然科学青年基金资助项目的资助。王家兵 博士,讲师;张 军 博士,教授;彭 宏 教授,博士生导师。

"http://www.…"来获得某种"财富"与帮助。开发一个功能强大的自动推理系统对开发者提出了较高的技术要求,有时更为重要的一点是,自动推理系统往往对硬件系统配置的需求较高。对许多用户而言,或者不具备开发自动推理系统的有关知识,或者不具备系统资源,但是又需要自动推理系统来完成他(她)的有关需求。在这种情形下,分布在 Internet 上的自动推理系统无疑是丰富的可用资源。此时,在自动推理 Web 服务器与 Web 客户间也涉及到消息(数据)的交换格式问题。显然,构造基于 Web 服务的自动推理系统对现有的系统测试与评价体系也是一个有益的补充:Web 客户可以对系统进行远程测试并得出自己的结论。

可扩展标记语言 XML[2](Extensible Markup Language)已成为在 Internet 上交换数据与文档的事实上的标准格式。自从提出 XML 以来,许多基于 XML 的应用已经出现,数学标记语言 MathML[3]也许是最著名的之一。

本文提出一个基于多种类逻辑的知识标记语言 MSKML(Many-Sorted Knowledge Markup Language)作为在 Web 上标记知识的语言。与现有流行的规则标记语言 RuleML、HornML[4,5]等相比,MSKML 具有如下特点:一方面,MSKML 的元素名直接对应于多种类逻辑与归结原理的有关术语,这样自动推理界可以很快熟悉 MSKML,也可以设计一个程序自动地将问题的子句集与反驳过程转换成与其对应的 XML 文档;另一方面,MSKML 不仅能够表示基于经典一阶逻辑的知识,也能表示基于多种类逻辑的知识。

为了给 Web 客户提供自动推理 Web 服务,实现了一个 COM(Component Object Model)对象 Prover,Prover 的实际推理过程采用基于 RLD[6](Rightmost Linear Deduction)演绎的多种类谓词演算。服务器与客户机之间采用简单对象通信协议 SOAP[7,8](Simple Object Access Protocol)作为网络通信协议。用 Web 服务描述语言 WSDL[9](Web Services Description Language)来对服务器提供的自动推理服务 ProverService 加以描述。

Web 客户需要自动推理 Web 服务时,首先把需要解决的问题表示成符合 MSKML 的 DTD(Document Type Definition,文档类型定义)的 XML 文件,然后作为一个 SOAP 请求消息的一部分发送给服务器;服务器把问题的解也表示成符合 MSKML 的 DTD 的一个 XML 文件并作为上述请求的响应消息的一部分返回给 Web 客户。

本文第 2 节通过一个实例简要介绍多种类逻辑以及 RLD(Rightmost Linear Deduction)演绎,第 3 节给出 MSKML 的 DTD,第 4 节给出自动推理 Web 服务的接口定义及其 Web 服务描述,第 5 节我们以著名的人工智能问题—Steamroller 问题—为例,给出该问题与自动推理 Web 服务给出的 RLD 反驳的 MSKML 表示,最后是有关结论。

2 多种类逻辑及 RLD 演绎

本文假设读者熟悉一阶逻辑及归结原理的基本术语,参见文[10,11]。

为了提高推理效率,人们提出了多种类逻辑及其谓词演算[12,13]。多种类逻辑的基本思想就是对定义域的元素进行分类,构成种类层次图,然后利用该层次的子类-父类关系对推理进行某种限制,从而加速推理过程。多种类逻辑已经在计算机科学的诸多领域得到了广泛运用,如知识表示[14]、程序设计语言[15]、软件规约与验证[16]等等。

与基于归结原理的经典谓词演算[10,11]相比,多种类谓词演算具有更高的演绎效率,参见文[12,13]。其中的主要原因是:一方面,当把同一个问题符号化为子句形式时,多种类逻辑具有较少的文字与子句数目;另一方面,多种类合一算法[17]的约束条件极大地缩小了搜索空间。关于这两点,我们可以通过例 1—著名的 Steamroller 问题[18]的简化版本—清楚地看出。

例 1 (1) 鸟与蛇都是动物。(2) 有一些鸟与蛇。(3) 有一些植物。(4) 每一种动物都喜欢吃所有那些不喜欢吃所有植物的动物。(5) 鸟不喜欢吃蛇。(6) 因此,一定有一种喜欢吃所有植物的动物。

我们用多种类逻辑来公理化这个问题:

谓词 $E(xy)$:x 喜欢吃 y。

$E(AA)$ $E(AP)$

sort:$B < A$ sort $S < A$ type:b : B

type:s : S type:p : P type:a_1 : A

type:a_2 : A type:p_1 : P type:b_1 : B

type:s_1 : S type:$g(A)$: P

上述表达式中,大写字母 A,B,S,P 是种类(Sort)符号,分别表示动物(Animal)、鸟(Bird)、蛇(Snake)及植物(Plant)。$E(AA)$与 $E(AP)$表达式表示二元谓词 E 的定义域(Domain Sort)为种类字符串 AA 或 AP,其中第一个参数的种类(Parameter Sort)为 A,第二个参数的种类为 A 或者 P。sort 表达式表示种类之间的关系,也称为种类层次图(Sort Hierarchy),这类似于面向对象程序设计语言中的继承关系,例如 sort:$B < A$,sort:$S < A$ 表示种类 B,S 均是 A 的子类,反之 A 是 B,S 的父类。

b,s,p 均为常量符号,a_1,a_2,b_1,s_1,p_1 均为变量符号。type 表达式类似于面向对象程序设计语言中的类型声明,例如 type:b : B 表示 b 是一个种类为 B 的常量,也称 b 的值域(Range Sort)为 B,type:b_1 : B 表示 b_1 是种类为 B 的变量。表达式 type:$g(A)$: P 表示一元 Skolem 函数 g 的定义域的种类为 A(g 是 Skolem 化逻辑公式时出现的,见后),值域的种类为 P,也即 g 带有一个种类为 A 的参数,且返回值的种类为 P。

上述谓词的定义域(常量、变量与函数的定义域及值域)也称为谓词的基调(常量、变量与函数的基调)(Signature)。

鸟不喜欢吃蛇:$(\forall b_1)((\forall s_1)((\neg E(b_1 s_1))))$

每一种动物都喜欢吃所有那些不喜欢吃所有植物的动物:

$(\forall a_1)(\forall a_2)(\forall p_1)(\neg E(a_2 p_1) \rightarrow E(a_1 a_2))$

因此一定有一种喜欢吃所有植物的动物:$(\exists a_1)((\forall p_1)(E(a_1 p_1)))$

我们再将上述逻辑公式转化为 Skolem 标准型及子句形式。

$(\forall a_1)(\forall a_2)(\forall p_1)(\neg E(a_2 p_1) \rightarrow E(a_1 a_2)) = (\forall a_1)(\forall a_2)(\forall p_1)(E(a_2 p_1) \vee E(a_1 a_2))$。

欲证结论的否:$\neg(\exists a_1)((\forall p_1)(E(a_1 p_1))) = (\forall a_1)((\exists p_1)(\neg E(a_1 p_1))) = (\forall a_1)(\neg E(a_1 g(a_1)))$。

因此我们有多种类逻辑的子句集如下({ }里的内容理解为析取形式,以下相同):

(1) $\neg E(b_1 s_1)$ (2)$\{E(a_2 p_1), E(a_1 a_2)\}$

(3)$\{\neg E(a_3 g(a_3))\}$

项具有种类不但减少了子句中的文字数目以及子句数目

(若用经典一阶逻辑来公理化例1,需要9个子句,且子句含有更多的文字),更重要的是它极大地缩小了搜索空间。譬如,若子句集(1)$\rightarrow E(b_1 s_1)$,(2)$\{E(a_2 p_1), E(a_1 a_2)\}$,(3)$\{\rightarrow E(a_3 g(a_3))\}$是经典一阶逻辑中的子句,则第1个子句的文字$\rightarrow E(b_1 s_1)$可以与第2个子句中的两个文字都可以进行归结。但在多种类谓词演算中,第1个子句的文字$\rightarrow E(b_1 s_1)$只可以与第2个子句中的文字$E(a_1 a_2)$进行归结,而不能与第2个子句中的文字$E(a_2 p_1)$进行归结。为什么呢,因为这两个文字的第二个参数的种类没有关系,前一个的种类是S,后一个的种类是P,而S和P之间没有子类-父类关系,因此这两个文字不能合一,从而不能归结;而在经典谓词演算中,由于谓词所带的参数没有种类,因此两个文字可以归结。

为了进一步提高基于归结原理的演绎效率及简化计算机程序设计结构,在文[6]中,我们提出了RLD演绎并证明了它的完备性定理。

在本文的余下部分,我们把一个子句看作是一个文字序列,即子句中的文字是有序的。子句的第一个文字称为最左文字,最后一个文字称为最右文字。如子句$C = \{P, Q, R\}$的最左文字为P,最右文字为R。同时,把两个子句C与D关于文字$L \in C$及$K \in D$的归结式$\sigma(C-\{L\}) \cup \sigma(D-\{K\})$看作是由文字序列$\sigma(C-\{L\})$与$\sigma(D-\{K\})$连接所得到的文字序列。如子句$C = \{P, Q, R\}$与$D = \{T, \rightarrow R\}$的归结式为$\{P, Q, T\}$(有序)。

定义1 对两个子句C,D及两个文字$L \in C$及$K \in D$,称由以下方式得到的子句为关于C的约化归结式:首先删去$\sigma(D-\{K\})$中与$\sigma(C-\{L\})$中某个文字相同的任何文字,得到子句$(\sigma(D-\{K\}))'$,然后将子句$(\sigma(D-\{K\}))'$连接在子句的$\sigma(C-\{L\})$后面得子句$\sigma(C-\{L\}) \cup (\sigma(D-\{K\}))'$,其中$\sigma \in \mathrm{mcu}_\Sigma\{|L|, |K|\}$。

例2 两个子句$C = \{P, Q, R\}$,$D = \{T, Q, \rightarrow R, P\}$关于$C$的约化归结式为$\{P, Q, T\}$而不是$\{P, Q, T, Q, P\}$。

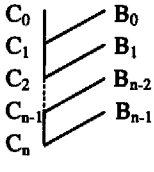


图1 RLD-演绎

定义2 设S是子句集,C_0是S中一个子句。从S出发以C_0为顶推出C_n的RLD演绎是如图1的一个演绎,其中:

1) 对于$i = 0, 1, \cdots, n-1$,C_{i+1}是C_i(称为中心子句)与B_i(称为边子句)关于C_i的约化归结式,且C_i中的归结文字是C_i中最右边的文字;

2) 每个B_i或者是S中子句,或者是某个C_j,$j < i$;

3) 演绎中无重言式。

定理1 设S是不可满足的子句集,C是S中一个子句。如果$S-\{C\}$是可满足的,则存在以C为顶从S推出空子句$\square$的RLD演绎[6]。

3 MSKML的文档类型定义DTD

一个XML文档的结构由该文档的DTD[2]或模式[19,20](Schema)所定义;而该文档则称为其相应的DTD或Schema的实例(Instance)。DTD源于标准通用标记语言SGML,采用上下文无关文法,使用简单。但DTD也有明显的缺陷,例如DTD的语法与XML文档的语法不同,所支持的数据类型不丰富等。为了克服DTD的弱点,人们提出了模式的概念。与DTD不同,XML文档的模式本身就是一个XML文档,且XML模式具有更丰富的内建数据类型,用户也可以根据需要用多种方式自己定义类型,例如可以像面向对象程序设计语言那样,以继承的方式从一个已有类型来创建新的类型等。因为篇幅所限,本文只给出MSKML的DTD,但根据DTD很容易将之转换成一个模式文件。

在MSKML中,有以下元素:〈kb〉,〈sort_hierarchy〉,〈signature〉,〈conjunction〉,〈conclusion〉,〈solution〉,〈disjunction〉,〈clause〉,〈resolvent〉,〈literal〉,〈posliteral〉,〈negliteral〉,〈pre_signature〉,〈fun_signature〉,〈var_signature〉,〈const_signature〉,〈fun_name〉,〈pre_name〉,〈var_name〉,〈const_name〉,〈domain_sort〉,〈parameter_sort〉,〈sort〉,〈sort_tree〉,〈range_sort〉,以及〈struct〉。

所有元素名均是自解释的。MSKML的元素名直接对应于多种类逻辑与归结原理的有关术语,这样,自动推理界很容易熟悉MSKML,也可以通过一个程序自动地将问题表示与推理过程转换成XML文档。一个知识基(〈kb〉)是一个子句集的合取形式(〈conjunction〉),一个子句(〈clause〉)是文字的析取形式(〈disjunction〉),一个文字(〈literal〉)可以是一个正文字(〈posliteral〉)或一个负文字(〈negliteral〉)。一个种类层次(〈sort_hierarchy〉)是一个由种类树结构(〈sort_tree〉)构成的森林结构。基调(〈signature〉)由谓词基调(〈pre_signature〉)、函数基调(〈fun_signature〉)、常数基调(〈const_signature〉)及变量基调(〈var_signature〉)构成。谓词基调由谓词名(〈pre_name〉)与参数的定义域构成,参数的定义域(〈domain_sort〉)是一个种类的字符串,由一系列参数种类(〈parameter_sort〉)来构成。函数(常量除外)基调(〈fun_signature〉)由函数名(〈fun_name〉)、参数定义域及函数的值域(〈range_sort〉)构成,常数基调(〈const_signature〉)由常数名〈const_name〉与常数的值域构成,变量基调(〈var_signature〉)由变量名(〈var_name〉)与变量的值域构成。〈struct〉用来递归定义项。

因为一个归结式(〈resolvent〉)是由两个父子句演绎而来,因此每个子句具有ID属性,以便在归结式中引用。归结式(〈resolvent〉)是一个由两个子句演绎出的子句,因此它有两个引用属性parent_fst与parent_snd,表示该归结式的两个父子句。一个问题的解(〈solution〉)是一个归结式的序列。元素〈conclusion〉的设立是基于如下考虑:基于归结方法的自动推理系统大多是采用反驳法,即若想从子句集S推出结论C,则要设法从子句集$S \cup \rightarrow C$推出空子句$\square$。为了把欲证结论的非与其它子句区分开来,我们用元素〈conclusion〉来表示它。

MSKML的DTD具体定义如下:

```
〈! ELEMENT kb (sort_hierarchy, signature, conjunction, conclusion, solution) 〉
〈! ELEMENT sort_hierarchy (sort | sort_tree) * 〉
〈! ELEMENT signature (pre_signature *, fun_signature *, const_signature *, var_signature *) 〉
〈! ELEMENT conjunction (clause+) 〉
〈! ELEMENT conclusion (conjunction) 〉
〈! ELEMENT solution (resolvent *) 〉
〈! ELEMENT sort_tree (sort | sort_tree)+ 〉
〈! ELEMENT pre_signature (pre_name, domain_sort) 〉
〈! ELEMENT fun_signature (fun_name, domain_sort, range_sort) 〉
〈! ELEMENT const_signature (const_name, range_sort) 〉
〈! ELEMENT var_signature (var_name, range_sort) 〉
〈! ELEMENT clause (disjunction) 〉
```

```
<! ATTLIST clause id ID #REQUIRED>
<! ELEMENT resolvent (disjunction) >
<! ATTLIST resolvent id ID #REQUIRED parent_fst IDREF #
REQUIRED parent_snd IDREF #REQUIRED>
<! ELEMENT disjunction (literal*) >
<! ELEMENT domain_sort (parameter_sort+)>
<! ELEMENT literal (posliteral | negliteral) >
<! ELEMENT posliteral (pre_name, (const_name | var_name |
struct)+) >
<! ELEMENT negliteral (pre_name, (const_name | var_name |
struct)+) >
<! ELEMENT struct (fun_name, (const_name | var_name |
struct)+) >
<! ELEMENT sort (#PCDATA)>
<! ELEMENT parameter_sort (#PCDATA)>
<! ELEMENT range_sort (#PCDATA)>
<! ELEMENT pre_name (#PCDATA) >
<! ELEMENT fun_name (#PCDATA) >
<! ELEMENT const_name (#PCDATA) >
<! ELEMENT var_name (#PCDATA) >
```

从 MSKML 的 DTD 定义可以看出，虽然 MSKML 是一个基于多种类逻辑的知识标记语言，但同时也是基于经典一阶逻辑的知识标记语言。当表达种类层次与基调的元素〈sort_hierarchy〉及〈signature〉在 MSKML 的 DTD 实例中为空元素时，该实例即为基于经典一阶逻辑的知识标记。

4 Web 服务接口定义及 Web 服务描述

4.1 接口定义

简单对象访问协议 SOAP(Simple Object Access Protocol)[7,8]是基于 XML 的用在分布式环境中交换信息的标准协议。SOAP 以 XML 形式提供了一个简单的用于在分布式环境中交换结构化和类型化信息的机制。SOAP 通过提供一个有标准组件的包模型和在模块编码数据的机制，定义了一个简单的表示应用程序语义的机制。这使 SOAP 能够被用于从消息传递到远程过程调用(RPC)的各种系统。有 SOAP 标准之后，所有的组件模型、开发工具、程序语言和应用系统都可以使用 SOAP 来沟通和交换信息，因此再也不需要任何特定技术来作为中介的转换机制。

SOAP Toolkit[21]是提供给程序开发人员用来创建、发送与接受以及处理 SOAP 消息的 SOAP 应用程序编程接口。SOPA Toolkit 3.0 支持以附件的形式发送或接受 XML 文件。我们用 Visual C++ 6.0 及 Microsoft SOAP Toolkit 3.0 实现了一个 COM 对象 Prover，Prover 有一个接口 IProver。接口 IProver 的实现类实现了自动推理 Web 服务 ProverService。自动推理 Web 服务 ProverService 的实际推理过程采用使用 RLD 演绎的多种类谓词演算。

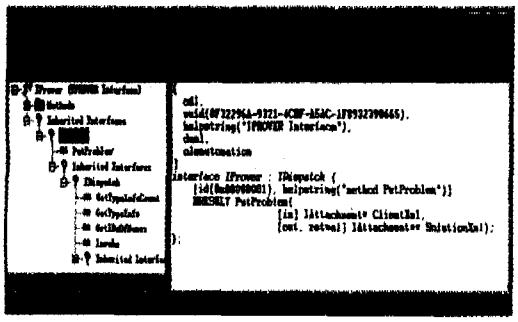


图 2 接口 IProver 的有关信息

接口 IProver 有一个方法 PutProblem，其原型如下：

HRESULT PutProblem([in] IAttachment * ClientXml, [out, retval] IAttachment * * SolutionXml)

PutProblem 带有两个参数：ClientXML 与 SolutionXML，它们都是 IAttachment 类型(SOAP Toolkit 3.0 支持的附件类型)，其中 ClientXML 是输入参数，而 SolutionXML 是输出参数。当 Web 用户需要远程调用 Web 服务 ProverService 时，首先把问题表示成符合 MSKML 模式的 XML 文件存储在 ClientXML 中，并以附件的形式作为 SOAP 请求消息发送给服务器；而服务器将所发现的该问题的解表示成符合 MSKML 模式的 XML 文件并存储在 SolutionXML 中，然后以附件的形式作为 SOAP 应答消息发送给客户端。

接口 IProver 的有关信息用微软的类型库查看工具 OLE View 显示在图 2。

4.2 Web 服务描述的 WSDL 文件

Web 用户在实际调用 Web 服务以前还必须知道该 Web 服务的有关信息，例如该 Web 服务具有哪些方法可以调用，每个方法的参数个数及类型是什么，等等。这些信息都由该 Web 服务的 WSDL 文件来描述，并放在服务器端，任何需要该 Web 服务的用户都可以从服务器获得该 WSDL 文件，并按照 WSDL 文件所描述的信息来格式化一个 SOAP 请求消息，并发往服务器。

WSDL 是由 IBM、微软等多家公司开发的，现以进入 W3C 标准的讨论阶段[9]。其宗旨是使用一个标准的输出接口来定义 Web 服务的实现程序代码提供的功能，以便让 Web 用户可以通过这个标准的输出接口来调用该服务。描述一个 Web 服务的 WSDL 文档是一个以 XML 组成的文件，这个文件内容叙述了实现程序代码对外提供的函数原型，也就是各种可供调用的函数名称以及参数消息。在实现程序代码提供了 WSDL 后，客户端就可以通过 WSDL 来调用实现程序代码提供的服务。WSDL 文档就像一个服务器与客户机之间的契约，服务器为客户提供请求的服务当且仅当该客户机的请求消息格式符合 WSDL 文档描述的要求。

因篇幅的关系，ProverService 的 WSDL 文件描述略。

5 应用

在本节，我们以著名的人工智能问题—Steamroller 问题—为例，用 Prover 系统给出该问题的 RLD 反驳，并用 MSKML 语言来表示该问题及 RLD 反驳。

1978 年 Alberta 大学的 L. Schubert 教授提出了以下具有挑战性的、著名的人工智能问题[18]：

狼(wolves)、狐狸(foxes)、鸟(birds)、毛虫(caterpillars)及蛇(snails)都是动物(animals)，且存在着这些动物。同时也存在着一些谷物(grains)，且谷物是一种植物(plants)。每一种动物喜欢吃所有的植物或所有那些比本身小且喜欢吃某些植物的动物。毛虫和蛇比鸟小，鸟又比狐狸小，狐狸又比狼小。狼不喜欢吃狐狸和谷物，鸟喜欢吃毛虫而不喜欢吃蛇，毛虫与蛇喜欢吃某些植物。因此，有一种动物喜欢吃某种喜欢吃谷物的动物。

尽管这个问题看似简单，然而许多证明器对它无能为力。并且它的搜索空间实在太大，即使手工计算也不容易。本节，我们运用 RLD 演绎来解决该问题。

Steamroller 问题描述的多种类子句集如下[18]：

type w ; W type f ; F type b ; B
type c ; C type s ; S type g ; G
type h(C) ; P* type i(S) ; P* type j(AA) ; G*
sort W < A sort F < A sort B < A
sort C < A sort S < A sort G < P
(1){E($a_1 p_1$), ¬M($a_2 a_1$), ¬E($a_2 p_2$), E($a_1 a_2$)}
(2){M($c_1 b_1$)}(3){M($s_1 b_1$)}
(4){M($b_1 f_1$)}(5){M($f_1 w_1$)}
(6){¬E($w_1 f_1$)}(7){¬E($w_1 g_1$)}
(8){E($b_1 c_1$)} (9){¬E($b_1 s_1$)}

(10){E(c_1h(c_1))}(11){E(s_1i(s_1))}
(12){¬E(a_1 a_2), (E(a_2j(a_1 a_2))}

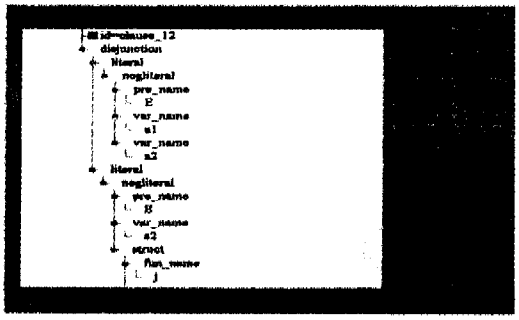


图 3　子句集在客户端的 MSKML 表示

Web 用户需要自动推理服务时，可以用任何程序设计语言编写客户端应用程序，把所需解决的问题表示成符合 MSKML 的 DTD 的 XML 文件并作为 SOAP 请求消息的一部分发送给自动推理服务器，服务器将问题的解也编码成符合 MSKML 的 DTD 的 XML 文件并作为 SOAP 请求消息的应答消息发送给客户。我们用 Visual C^{++} 6.0 及 SOAP Toolkit 3.0 开发了一个客户端程序 ProverClient，并用 Steamroller 问题对该服务器进行了测试。Steamroller 问题的 MSKML 表示在客户端的显示如图 3。

在图 3 中，嵌入在 id 属性值为“clause_12”中的内容对应于子句集中的子句(12)，即我们欲证结论的否定。

Prover 给出的 Steamroller 问题的一个 RLD 反驳如下：

(1){E(a_1 p_1), ¬M(a_2 a_1), ¬E(a_2 p_2), E(a_1 a_2)};
(13){E(b_1 p_1), ¬M(s_1 b_1), ¬E(s_1 p_2)}，子句 1 与子句 9 的归结式；
(14){E(b_1 p_1), ¬M(s_1 b_1)}，子句 13 与子句 11 归结式；
(15){E(b_1 p_1)}，子句 14 与子句 3 的归结式；
(16){¬E(a_1 b_1)}，子句 15 与子句 12 的归结式；
(17){E(a_1 p_1), ¬M(b_1 a_1), ¬E(b_1 p_2)}，子句 16 与子句 1 的归结式；
(18) {E(a_1 p_1), ¬M(b_1 a_1)}，子句 17 与子句 15 的归结式；
(19){E(f_1 p_1)}，子句 18 与子句 4 的归结式；
(20){¬E(a_1 f_1)}，子句 19 与子句 12 的归结式；
(21){E(a_1 p_1), ¬M(f_1 a_1), ¬E(f_1 p_2)}，子句 20 与子句 1 的归结式；
(22) {E(a_1 p_1), ¬M(f_1 a_1)}，子句 21 与子句 19 的归结式；
(23){E(w_1 p_1)}，子句 22 与子句 5 的归结式；
(24)□，子句 23 与子句 7 的归结式。

在上述 RLD 反驳中，除了顶子句(1)是问题本身给定的以外，其它每一个子句都是它前面的子句与某一个子句的 RLD 归结式，其中子句(24)是一个空子句，即我们用 RLD 演绎证明了该问题。

用自然语言来描述这个问题的证明过程如下：

(1) 每一种动物喜欢吃所有的植物或喜欢吃所有那些比本身小且喜欢吃某些植物的动物；(给定的条件，即子句 1)

(13) 因为鸟不喜欢吃蛇(子句 9)，所以鸟喜欢吃所有的植物或蛇不比任意的鸟小或蛇不喜欢吃任何植物；

(14) 因为蛇喜欢吃某些植物(子句 11 及 skolem 函数 i)，所以鸟喜欢吃所有的植物或蛇不比任意的鸟小；

(15) 因为蛇比鸟小(子句 3)，所以鸟喜欢吃所有的植物；

(16) 因为不存在这样一种动物，它喜欢吃某种喜欢吃谷物的动物(子句 12，即欲证结论的否定)，而谷物是一种植物，所以没有动物喜欢吃鸟；

(17) 因为每一种动物喜欢吃所有的植物或喜欢吃所有那些比本身小且喜欢吃某些植物的动物(子句 1)，所以每一种动物喜欢吃所有的植物或鸟不比任何动物小或鸟不喜欢吃任何植物；

(18) 因为鸟喜欢吃所有的植物(子句 15)，所以每一种动物喜欢吃所有的植物或鸟不比任何动物小；

(19) 因为鸟比狐狸小(子句 4)，所以每一种动物喜欢吃所有的植物，而狐狸是动物，所以狐狸喜欢吃所有的植物；

(20) 因为不存在这样一种动物，它喜欢吃某种喜欢吃谷物的动物(子句 12)，而谷物是一种植物，所以没有动物喜欢吃狐狸；

(21) 因为每一种动物喜欢吃所有的植物或喜欢吃所有那些比本身小且喜欢吃某些植物的动物(子句 1)，所以每一种动物喜欢吃所有的植物或狐狸不比任意动物小或狐狸不喜欢吃任何植物；

(22) 因为狐狸喜欢吃所有的植物(子句 19)，所以每一种动物喜欢吃所有的植物或狐狸不比任意动物小；

(23) 因为狐狸比狼小(子句 5)，所以每一种动物喜欢吃所有的植物，而狼是动物，所以狼喜欢吃所有的植物；

(24) 因为狼不喜欢吃谷物(子句 7)，而谷物是一种植物，所以狼不喜欢吃某些植物，矛盾。

自动推理 Web 服务 ProverService 返回给客户端的 RLD 反驳在客户端的显示如图 4。

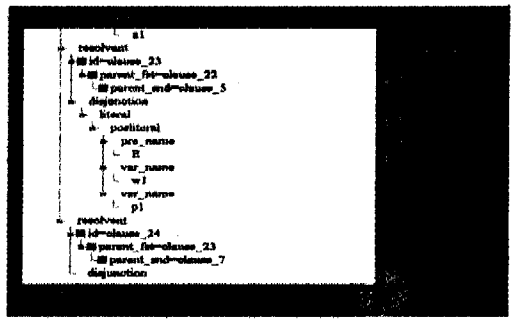


图 4　RLD 反驳在客户端的 MSKML 表示

在图 4 中，嵌入在〈resolvent〉且 id 属性值为“clause_24”的只含一个空的〈disjunction〉元素，它对应于 RLD 反驳中的子句(24)，即子句(23)与子句(7)的归结式，是一个空子句。

结论　Internet 环境下的知识表示与自动推理是广大人工智能及相关领域研究者面临的新课题。本文对此进行了一定的探索，所作工作总结如下：

(1) 为了给自动推理领域提供一个统一的问题(知识)表示格式，提出了一个基于多种类逻辑的知识标记语言 MSKML，给出了 MSKML 标记语言的 DTD 的详细定义。

(2) MSKML 的元素名直接对应于多种类逻辑与归结原理的有关术语，这样做的优点是：一是自动推理界可以很快熟悉 MSKML，二是可以设计一个程序自动地将问题的子句集与反驳过程转换成与其对应的 XML 文档。

(3) 与现有流行的规则标记语言 RuleML、HornML 相比，MSKML 不仅能够表示基于经典一阶逻辑的知识，也能表示基于多种类逻辑的知识。

(4) 实现了 COM 对象 Prover 来提供自动推理 Web 服务 ProverService。Prover 的实际推理过程采用基于 RLD 演绎的多种类谓词演算。ProverService 采用简单对象访问协议 SOAP 作为服务器与 Web 客户间的网络通信协议，用 Web 服务描述语言 WSDL 对其提供的服务进行了描述。

(5) 给出了著名的人工智能问题—Steamroller 问题—的 RLD 解，并以它为例，给出了该问题及其 RLD 反驳的 MSKML 表示以及自动推理 Web 服务的调用过程。

参 考 文 献

1 Sutcliffe G, Suttner C. Evaluating general purpose automated theorem proving systems. Artificial Intelligence, 2001, 131(1): 39～54

2 Bray T, Paoli J, Sperberg-McQueen C M. Extensible Markup Language (XML) 1.0, W3C Recommendation, February 1998. http://www.w3.org/ TR/1998/REC-xml-19980210

3 Carlisle D, Ion P, Miner R, et al. Mathematical Markup Language (MathML) Version 2.0, 2001. http://www.w3.org/TR/2001/REC-MathML2-20010221/

4 RuleML, HornML. http://www.dfki.uni-kl.de/ruleml

5 RuleML Initiative. http://www.dfki.de/ruleml

6 王家兵，徐正权，王能超. RLD演绎及子句蕴含与子句包含关系的非等价性. 计算机研究与发展，2002，39(12)：1630～1636

7 Gudgin M, Hadley M, Mendelsohn N, et al. SOAP Version 1.2 Part 1: Messaging Framework, W3C Candidate Recommendation, December 2002. http://www.w3.org/TR/2002/CR-soap12-part1-20021219

8 Gudgin M, Hadley M, Mendelsohn N, et al. SOAP Version 1.2 Part 2: Adjuncts, W3C Candidate Recommendation, Dec. 2002, http://www.w3.org/TR/2002/CR-soap12-part2-20021219

9 Chinnici R, Gudgin M, Moreau J-J, et al. Web Services Description Language (WSDL) Version 1.2, W3C Working Draft, March 2003. http://www.w3.org/TR/2003/WD-wsdl12-20030303

10 Robinson J A. A machine-oriented logic based on the resolution principle. J. ACM, 1965, 12(1): 23～41

11 刘叙华. 基于归结方法的自动推理. 北京：科学出版社，1994

12 Walther C. A Many-Sorted Calculus Based on Resolution and Paramodulation. London: Pitman / Los Alamitos: Morgan Kaufmann, 1987

13 Cohn A G. A more expressive formulation of many sorted logic. J. Automated easoning, 1987, 3: 113～200

14 Beierle C, Hedtstück U, Pletat U, et al. An order-sorted logic for knowledge representation systems. Artificial Intelligence, 1992, 55: 149～191

15 Goguen J A, Meseguer J. Order-sorted algebra I: equational deduction for multiple inheritance, overloading, exception and partial operations. Theoretical Computer Science, 1992, 105: 217～273

16 Weibel T. An order-sorted resolution in theory and practice. Theoretical Computer Science, 1997, 185(2): 393～410

17 Walther C. Many-sorted unification. J. ACM, 1988, 35(1): 1～17

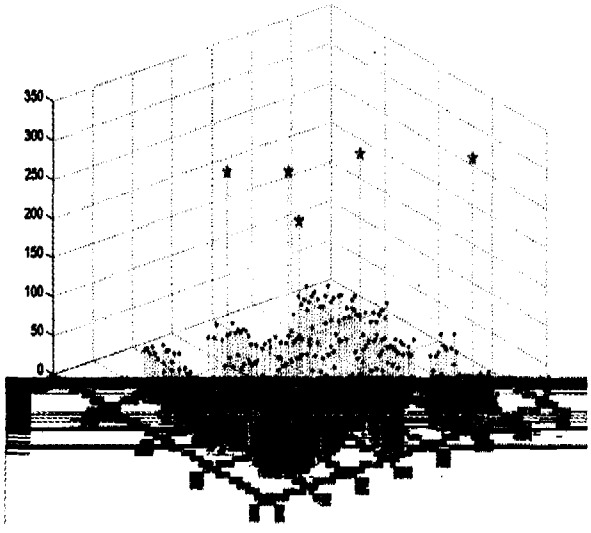
18 Walther C. A mechanical solution of Schubert's steamroller by many-sorted resolution. Artificial Intelligence, 1986, 26(2): 217～224

19 Thompson H S, Beech D, Maloney M, et al. XML Schema Part 1: Structures. W3C Recommendation, May 2001. http://www.w3.org/TR/2001/ REC-xmlschema-1-20010 502/

20 Biron P V, Malhotra A. XML Schema Part 2: Datatypes. W3C Recommendation, May 2001. http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/

21 SOAP Toolkit 3. http://msdn.microsoft.com/xml/, June 2002

(上接第 187 页)



i	50	80	150	210	260
Xi	320	300	280	290	330

图 1 仿真实验的结果

算法在实现时，计算 k 个最近邻居的算法是基于空间数据点的欧氏距离的，其他的均按照上述算法描述实现。运行结果如表 1 所示：将 $d^2(X_i)$ 的值按照降序排列，给出排名前 6 位的情况：

表 1 孤立点检测的运行结果

Xi	X260	X50	X80	X210	X150	X185
$d^2(X_i)$	71.6111	70.1971	51.5573	49.0483	43.1441	7.8590

由上表的结果可知，所检测出的五个孤立点分别为 260，50，80，210，150，与有意设定的五个孤立点相吻合，充分说明了算法的可行有效性。

结束语 文中给出了基于 Mahalanobis 距离的一种空间孤立点检测算法，实验结果表明该方法算法是可行有效的，同时分析了中心算法的时间复杂度。

空间孤立点检测除了上述研究的类型外还有时序数据和时空数据的孤立点检测，这些都涉及到其他的邻居的属性值。后续的工作研究就是进行时空数据的孤立点检测。

参 考 文 献

1 Berchtold S. The pyramid-technique: Towards breaking the curse of dimensionality. In: Proc. ACM SIGMOD Intl. Conf. on Management of Data. ACM Press. 1998. 142～153

2 Luc A. Exploratory Spatial Data Analysis and Geographic Information Systems. In: M. Painho ed. New Tools for Spatial Analysis, 1994. 45～54

3 Shekhar S, Lu C, Zhang P. Detecting Graph-Based Spatial Outlier: Algorithms and Applications (ASummary of Results). In Proc. of the Seventh ACMSIGKDD Int'l Conference on Knowledge Discovery and Data Mining. Aug 2001

4 Shekhar S, Lu C, Zhang P. Detecting Graph-Based Spatial Outlier. Intelligent Data Analysis: An International Journal, 2002. 451～468

5 Panatier Y, Variowin. Software For Spatial Data Analysis in 2D. New York: Springer-Verlag, 1996