从 *ALC* 到 *SHOQ*(*D*):描述逻辑及其 Tableau 算法

梅 婧 林作铨
(北京大学信息科学系 北京 100871)

摘 要 描述逻辑是一类知识表示的形式系统,并成为语义 Web 的逻辑基础。Tableau 是描述逻辑的基本证明论,基于 Tableau 的算法提供了描述逻辑的推理机。本文系统地阐述了对应于语义 Web 语言从基本的 ALC 到 SHOQ(D)的描述逻辑基础及其相应的 Tableau 算法。

关键词 描述逻辑,Tableau-算法,语义 Web

From *ALC* to *SHOQ*(*D*):A Survey of Tableau Algorithms for Description Logics

MEI Jing LIN Zuo-Quan
(Information Science Department, Peking University, Beijing 100871)

Abstract Description logics are a family of knowledge representation formalisms which become logical foundation of semantic Web. Tableaux are basic proof theories for description logics. Tableau-based algorithms provide the reasoning engines for description logics. In this paper, we survey various description logics and the corresponding tableau algorithms with respect from *ALC* to *SHOQ*(*D*) of the semantic Web languages.

Keywords Description logic, Tableau algorithm, Semantic Web

1 引言

描述逻辑是一类知识表示的形式系统,通过定义应用领域的概念及其结构关系,刻画领域内的个体信息[1]。

描述逻辑建立在概念和角色之上,由构子(constructor)从简单概念和角色构造出复杂概念和角色。概念对应于逻辑中的一元谓词,角色对应于二元谓词,构子决定着语言的表达能力,类似于逻辑联结词的功能。

ALC 是最基本的一种 DL 语言,构子包括合取、析取、否定、存在性限定和值限定[2]。在 *ALC* 的基础上,新增数量限定,函数性约束或定性数量限定,就分别演变为 *ALCN*,*ALCF*,*ALCQ*。

实际应用中(如数据库、语义 Web),不仅描述概念,还要增强角色的表达能力。具有传递性的角色常用于构造复合对象(如"祖先""子孙"),ALC_{RA+}是在 *ALC* 基础上允许部分角色具有传递性[3,4]。参照这种形式语言与命题(多)模态逻辑 S_4 的紧密联系[5],简称其为 *S*。进一步,若纳入角色包含公理(如"父子关系"⊑"家长-孩子关系")形成角色分层则得到 *SH* 语言。另一方面,若 *S* 中对角色的逆是封闭的,即存在"逆角色"算子,那么 *SI* 随即形成。既能通过整体刻画部分,亦能由部分描述整体,这就是 *SHI* 的独特之处。当然在 *SHI* 的基础上再添加数量限定、函数性约束或定性数量限定,自然就有了 *SHIN*,*SHIF*,*SHIQ*[6~9]。

当需要对个体实例进行刻画,通过枚举实例来描述一个概念时(如,联合国常任理事国={中国,法国,俄罗斯,英国,美国}),可在描述逻辑中加入"集合"算子(也称枚举算子 one-Of)将一些个体名整合成一个集合概念。如 *SHOQ*,*SHOIN* 就是在 *SHQ*,*SHIN* 的基础上扩展而来[11]。除了描述抽象概念,实际应用中还常常需要对诸如整数、字符串之类的具体数据类型和数据值进行刻画和推理。因此包括有型域(concrete domain)[12],在 *SHIF*,*SHOIN*,*SHOQ* 基础上构建了现今较为关注的形式系统,如 *SHIF*(*D*),*SHOIN*(*D*),*SHOQ*(*D*)等。

Tableau 演算是一种一阶逻辑的证明论。一般地,描述逻辑是一阶逻辑的一个可判定子集,因而能够构造出可靠完全的 Tableau 算法,用于判定系统的推理问题。描述逻辑的 Tableau 算法最早由 Schmidt-Schauβ 和 Smolka 为检验 *ALC*-概念的可满足性而提出[3],被广泛用于各种描述逻辑中以判定概念的可满足性或概念间的包含关系[7,9,10]。各种优化的 Tableau 算法已在实用推理机中得以实现[21,22]。

语义 Web 的发展,使得描述逻辑备受关注。Tim Berners-Lee 等人于 1999 年提出的语义 Web 作为 Web 的扩展,其目的是通过结构化形式表示,将 Web 的内容融入其表示结构中,使得计算机程序能够对网络资源进行分析、处理,最终实现自动推理[14,15]。语义 Web 体系结构中[20],本体(Ontology)层主要用于描述网络资源及资源间的关联关系,是语义 Web 研究的重心[17,18]。OWL(Web Ontology Language)是由 W3C 最新规范的一种语义 Web 的本体语言[19],以描述逻辑为基础理论,在语法和语义中,都融合了 DL 的构建思想,尤其是 OWL Lite 和 OWL DL 可多项式归约到 *SHIF*(*D*)和 *SHOIN*(*D*)[16]。描述逻辑推理机 FaCT[21]和 RACER[22]已实现 OWL Lite 推理,但对于 SHOIN(D)[16]尚无实用算法,因此 OWL DL 的推理实现还需进一步研究。

本文根据语义 Web 本体语言应用背景,从较简单 *ALC* 到较复杂的 *SHOQ*(*D*),逐一展开讨论它们对应的描述逻辑基础及其 Tableau 算法。第 2 节介绍描述逻辑语言 *ALC* 的语法、语义及其 Tableau-算法,同时指出概念的可满足性能够由该可靠完全的算法判定,其计算复杂度为 PSPACE-完全的。第 3 节是 S-系列,包括 *SI*,*SHI*,*SHIF*,*SHIQ* 四部分,而 *SHOQ*(*D*)在第 4 节中详细介绍。理论上已证明,即使在最基

梅 婧 博士研究生,主要研究领域为语义 Web 与描述逻辑。林作铨 教授,博士生导师,主要研究领域为计算机科学和人工智能。

本的 ALC 上扩展逆角色算子(I)和个体枚举算子(O),同时包括有型域(D),算法复杂度也是不尽人意的[12,13],因此,在3.3和3.5节中详细介绍 $SHIF$ 和 $SHIQ$,在4.1和4.2节提出 O 和 D 后,我们选择 $SHOQ(D)$[11]作为对语义 Web 中描述逻辑的 Tableau 算法的介绍。事实上,语义 Web 上本体语言是顺势发展而来的,最新的 OWL 的前身是 OIL(the Ontology Inference Layer),而 OIL 的语法正对应着 $SHOQ(D)$ 的公理[18],因此,借鉴 $SHOQ(D)$ 不但有助于深刻理解基于 $SHIF(D)$ 的 OWL Lite,而且能促进基于 $SHOIN(D)$ 的 OWL DL 的研究。除了依次叙述这些形式系统的语法、语义及其各自的 Tableau-算法,本文还将着重阐述在逐步扩充系统的表达能力时,Tableau 算法是如何调整以保证算法的可靠完全性,且有一个较为合理的计算复杂度。

2 ALC

描述逻辑的主要表现手段是概念描述。一般由一组概念名和角色名,借助概念构子递归定义概念描述。构子决定了 DL 形式语言的表达能力。

ALC(attributive concept description language with complements)[3]是最基本的一种 DL 形式语言,构子仅有合取、析取、否定、存在性限定和值限定。下面主要论述其语法语义及 Tableau-算法,该算法用于判定 ALC-概念的可满足性,是可靠完全的,且属于 PSPACE-完全的计算复杂度类。

2.1 语法和语义

定义 2.1 设 N_C 是概念名的一个集合,N_R 是角色名的一个集合。定义 ALC-概念的集合为满足下列条件的最小集合:

1. 若概念名 $C\in N_C$,则 C 是 ALC-概念;

2. 若 C,D 是 ALC-概念,$R\in N_R$ 是角色名,则 $(C\sqcap D)$,$(C\sqcup D)$,$(\neg C)$,$(\forall R.C)$,$(\exists R.C)$ 都是 ALC-概念。

定义 2.2 ALC-解释 $I=(\Delta^I,\cdot^I)$ 由非空集合 Δ^I 和函数 $\cdot^I$ 组成。称 Δ^I 为 I 的论域。函数 $\cdot^I$ 将每个概念映射为 Δ^I 的一个子集,将每个角色映射为 $\Delta^I\times\Delta^I$ 的一个子集。同时满足下列语义:

$(C\sqcap D)^I=C^I\cap D^I$

$(C\sqcup D)^I=C^I\cup D^I$

$(\neg C)^I=\Delta^I\backslash C^I$

$(\forall R.C)^I=\{x\in\Delta^I\mid$对任意 $y\in\Delta^I$,

如果$\langle x,y\rangle\in R^I$,那么 $y\in C^I\}$

$(\exists R.C)^I=\{x\in\Delta^I\mid$存在$-y\in\Delta^I$,

满足$\langle x,y\rangle\in R^I$,而且 $y\in C^I\}$

定义 2.3 (1)称概念 C 是可满足的当且仅当存在解释 I 使得 $C^I\neq\emptyset$,I 称为 C 的模型。(2)称概念 D 包含概念 C 当且仅当对所有解释 I 都成立 $C^I\subseteq D^I$,记做 $C\sqsubseteq D$,(3)称概念 C,D 是等价的当且仅当 C,D 互包含即:$C\sqsubseteq D$,$D\sqsubseteq C$,记做 $C\equiv D$。(4)对于一个解释 I,称个体 $x(x\in\Delta^I)$ 是概念 C 的实例当且仅当 $x\in C^I$。

由于存在否定算子,我们可以将包含关系的推理归演为可满足性的证明,这是因为:$C\sqsubseteq D$ 成立当且仅当 $C\sqcap\neg D$ 是不可满足的。反之,可满足性也可演算为包含关系的证明,这是因为:C 是可满足的当且仅当 $C\sqsubseteq(P\sqcap\neg P)$ 不成立,其中 P 是任一概念名。因而,在下面的讨论中,我们只涉及概念的可满足性判定问题。

2.2 Tableau-算法

ALC 的 Tableau-算法是通过构造概念表达式 D 的模型来证明 D 的可满足性。称这个构造的模型为一棵完整树,模型中的个体对应着树的结点。完整树上结点的标注是一组 ALC-概念集合,边的标注是某个 ALC-角色名。

若要证明 ALC 概念 D 的可满足性,我们关注的是概念 D 的子概念集合 $sub(D)$,所以不妨将树上标注结点的概念集合限定为 $sub(D)$ 的子集,同时树上标注边的角色名也限定为 $sub(D)$ 中出现的角色名。

为了简化构造,我们把所有概念都转化为否定范式(negation normal form,NNF),即否定只出现在概念名前。通过 DeMorgan 法则,这种转化是易实现的。

明确好上述前提后,我们首先给出 ALC-表的形式定义,继而介绍 ALC Tableau-算法,并说明其算法性质。

2.2.1 ALC-表

定义 2.4 设 D 是一个 ALC-概念(D 是否定范式),R_D 是出现在 D 中的角色集合,$R_D=\{R\mid R$ 出现在 D 中$\}$。定义 D 的 ALC-表 T 为一个三元组(S,L,ε):

S:个体的集合;

$L:S\to 2^{sub(D)}$ 将 S 中的个体映射为 $sub(D)$ 的子集;

$\varepsilon:R_D\to 2^{S\times S}$ 将 R_D 中的角色映射为个体-个体对的集合,同时要求存在某个个体 $s\in S$ 使得 $D\in L(s)$。

对任意 $s\in S,C,C_1,C_2\in sub(D);R\in R_D$,$ALC$-表有下列五个性质:

(P1) 如果 $C\in L(s)$,那么:$\neg C\notin L(s)$

(P2) 如果 $C_1\sqcap C_2\in L(s)$,那么 $C_1\in L(s)$ 而且 $C_2\in L(s)$

(P3) 如果 $C_1\sqcup C_2\in L(s)$,那么 $C_1\in L(s)$ 或者 $C_2\in L(s)$

(P4) 如果$\forall R.C\in L(s)$且$\langle s,t\rangle\in\varepsilon(R)$,则 $C\in L(t)$

(P5) 如果$\exists R.C\in L(s)$,则存在某个 $t\in S$ 使得$\langle s,t\rangle\in\varepsilon(R)$且 $C\in L(t)$。

鉴于模型论与 Tableau-算法之间的紧密关系,可以:(1)通过 D 的表 T 构造 D 的模型 I;(2) 通过 D 的模型 I 构造 D 的表 T,继而由归纳法得证下面定理:

定理 2.1 一个 ALC-概念 D 是可满足的当且仅当存在 D 的一个 ALC-表 T。

事实上,由 D 的表 $T=(S,L,\varepsilon)$ 可如下构造 D 的模型 $I=(\Delta^I,\cdot^I)$:

$\Delta^I=S$

$CN^I=\{s\mid CN\in L(s)\}$

$R^I=\varepsilon(R)$

其中,CN 是所有在 $sub(D)$ 中出现的概念名。由于 T 是 D 的表,因此存在一个 $s_0\in S$ 使得 $D\in L(S_0)$。通过对概念的构建结构做归纳,可以证明:对于任一概念 E(不限于概念名 CN),如 $E\in L(s)$,那么 $s\in E^I$。因此得到 $S_0\in D^I$,所以 $D^I\neq\emptyset$。由此可知,I 是 D 的模型。

而由 D 的模型 $I=(\Delta^I,\cdot^I)$,也可如下构造 D 的表 $T=(S,L,\varepsilon)$。由概念 $(C\sqcap D)$,$(C\sqcup D)$,$(\neg C)$,$(\forall R.C)$,$(\exists R.C)$ 的语义解释可以直接得出表 T 满足性质 1～5,即 T 是 D 一个合法的 ALC-表。

$S=\Delta^I$

$L(s)=\{C\in sub(D)\mid s\in C^I\}$

$\varepsilon(R)=R^I$

2.2.2 构造 ALC-表

由定理 2.1 可知,对于 ALC-概念 D 如果存在算法能构造出其 ALC-表 T,那么就可以此判定 D 可满足性。运用算法规则(见表 1)扩展完整树的 Tableau-算法能实现这一目标。

完整树上的每个结点 x 都用一个集合 $L(x)$ 标注,$L(x)\subseteq sub(D)$;完整树上的每条边$\langle x,y\rangle$都用一个出现在 $sub(D)$ 中的角色名 R 标注,$L(\langle x,y\rangle)=R$。

表 1 *ALC*-Tableau-算法规则

规则名	规则
⊓-规则	如果 (1)$C_1 \sqcap C_2 \in L(x)$ (2)$\{C_1, C_2\} \not\subseteq L(x)$ 那么 $L(x) \to L(x) \cup \{C_1, C_2\}$
⊔-规则	如果 (1)$C_1 \sqcup C_2 \in L(x)$ (2)$\{C_1, C_2\} \cap L(x) = \emptyset$ 那么 $L(x) \to L(x) \cup \{C\}$, 对于某个 $C \in \{C_1, C_2\}$
∃-规则	如果 (1)$\exists S.C \in L(x)$ (2)x 没有一个 S-后续 y, 使得 $C \in L(y)$ 那么 新增一个结点 y,赋值 $L(\langle x,y\rangle) = S$ 且 $L(y) = \{C\}$
∀-规则	如果 (1)$\forall S.C \in L(x)$ (2)x 有一个 S-后续 y,但 $C \notin L(y)$ 那么 $L(y) \to L(y) \cup \{C\}$

只有当对形如$\exists R.C$的概念进行扩展时，才会在树上增加一条边，同时增加一个新结点。其余扩展都是在结点 x 的标注集合中增加新的概念。

如果结点 x 和结点 y 由一条边$\langle x,y\rangle$连接，即：$L(\langle x,y\rangle) = R$，那么称 y 为 x 的 R-后继(successor)。

如果结点 x 中既包括某个概念 C 又包括该概念的否定式$\neg C$，即：存在某个 C，有$\{C, \neg C\} \subseteq L(x)$，则称 $L(x)$中包含一个冲突(clash)。

Tableau-算法初始化树 T，仅由单结点 x_0组成，称 x_0 为树根。同时赋值 $L(x_0) = \{D\}$，D 就是待证可满足性的概念。通过不断应用算法规则(见表 1)，对树 T 进行扩展。

如果在完整树 T 上存在某个结点 x，$L(x)$ 中包含一个冲突，或者，对树 T 已没法实施任何算法规则了，那么称树 T 是完全的。

如果输入概念 D，应用算法规则后得到一棵完全的、无冲突的完整树，则 Tableau-算法将返回“概念 D 是可满足的”，否则返回“概念 D 是不可满足的”。

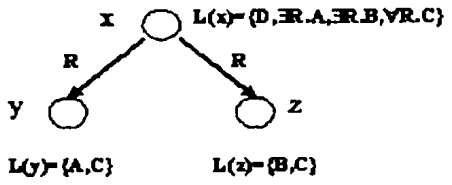


图 1 *ALC*-举例

举例说明，设待证可满足性的概念 $D = \exists R.A \sqcap \exists R.B \sqcap \forall R.C$，初始化完整树 T，树根 x 的结点标注为 $L(x) = \{D\}$。然后运用⊓-规则在结点 x 的标注中新增概念 $L(x) = \{D, \exists R.A, \exists R.B, \forall R.C\}$。再由∃-规则分别得到 x 的两个 R-后续 y 和 z，且有 $L(y) = \{A\}$和 $L(z) = \{B\}$。这时对 x 运用∀-规则将扩充 y 和 z 的结点标注，分别为 $L(y) = \{A, C\}$和 $L(z) = \{B, C\}$。由于已设有算法规则可供使用，得到一棵完全完整树(如图 1 所示)，而 x, y, z 三个结点标注中都没有包含冲突，所以概念 D 是可满足的。

2.2.3 *ALC* 的 Tableau-算法性质

定理 2.2[界限性] 对任一 *ALC*-概念 D，Tableau-算法都停机。事实上，算法的界限性取决于算法规则的特征，不会无止境地使用算法规则，因为：(1) 算法规则既没有从完整树上移走结点，也没有从结点标注中移走概念；(2) 只有形如$\exists R.C$的概念进行扩展时才产生后续结点，而且最多只能激发一个后续。所以完整树的分支个数被 D 中存在性限定算子的个数所界限，树的出度最大为$|sub(D)|$；(3) 由算法规则可知，沿着完整树的路径，结点标注内的概念个数是递减的，因此完整树的深度线性相关于输入概念 D 的大小，即深度最多为$|sub(D)|$。

定理 2.3[可靠性] 如果对 *ALC*-概念 D 应用算法规则得到一棵完全的、无冲突的完整树，那么 D 存在 *ALC*-表。事实上，如果 Tableau-算法对 D 构造了树 T，不妨如下定义表 $T = (S, L, \varepsilon)$：

$S = \{x | x$ 是树 T 上的结点$\}$

$L(x) = L(x)$

$\varepsilon(R) = \{\langle x,y\rangle \in S \times S | y$ 是 x 的 R-后续，

即：$L(\langle x,y\rangle) = R\}$

对于树 T 的根结点 x_0($x_0 \in S$)，在树的初始化时有 $D \in L(x_0)$，因而存在个体 $s \in S$ 使得 $D \in L(s)$。又由于 Tableau-算法构造的树 T 是完全的、无冲突的，因此可证明该表 T 满足 *ALC*-表的五个性质。

定理 2.4[完备性] 如果 *ALC*-概念 D 存在 *ALC*-表，那么根据 Tableau-算法而恰当运用算法规则，则可产生 D 的一棵完全的、无冲突的完整树。

事实上，借助概念 D 的 *ALC*-表 T 而恰当激活算法规则，由定理 2.2 知 Tableau-算法必终止，所以将得到一棵完全的完整树 T。同时由定义 2.4 *ALC*-表的性质 1 可知 T 也是无冲突的，否则存在某个 C，$\{C, \neg C\} \subseteq L(x) \subseteq L(\pi(x))$，这与性质 1 矛盾。其中 π 是将树 T 上的结点映射为 S 的元素，可证明对于所有结点 x 都满足 $L(x) \subseteq L(\pi(x))$。

定理 2.5[判定性] 对于 *ALC*-概念的可满足性和包含关系，Tableau-算法是可判定的。

已知可满足性与包含关系能够相互转化，根据以上各定理自然有 Tableau-算法的判定性。

定理 2.6[复杂度] *ALC*-概念的可满足性的判定问题是 PSPACE-完全的。

上述 *ALC* 的 Tableau-算法可能需要指数时间和空间，事实上由定理 2.2 已知完整树最大出度为$|sub(D)|$，最大深度为$|sub(D)|$，因而全部结点个数可达$|sub(D)|$ 的指数数量级。不过适当修改算法使之仅需多项式空间的优化算法是可行的：扩展完整树时，逐一处理各个分支，且深度优先遍历[3]。在运用算法规则时，尽可能先用⊔-规则和⊓-规则，并及时检查是否存在冲突；同时尽可能后用∃-规则产生后续结点。1970 年 Savitch 已证明：PSPACE=NPSPACE[23]，因此即使存在不确定的⊔-规则，也不会影响修改后的 Tableau-算法的多项式空间复杂度。

由于给定个体的后续结点能够逐个处理，因此算法仅需要存储生成的完整树的一条路径，即至多$|sub(D)|$个存储单元，这个多项式空间保证了 *ALC*-概念的可满足性的判定问题是属于 PSPACE 语言类。1991 年 Schmidt-Schauβ 和 Smolka 给出了从全量化布尔公式的判定问题(TQBF)到该问题的多项式时间归约[3]，所以该问题也是 PSPACE-难的。

3 S-系列

S 命名可追溯于命题(多)模态逻辑 S_4[5]，是在描述逻辑 *ALC* 的基础上允许部分角色具有传递性，即 ALC_R^+[3,4]。

本节首先在 S 中加入角色的逆算子，称为 SI 语言；然后纳入角色分层扩展成为 SHI；最后允许函数性约束和定性数量限定，分别形成 $SHIF$ 和 $SHIQ$ 语言。

3.1 SI

具有传递性的角色名集合 N_{R_+} 是全部角色名集合 N_R 的一个子集，任一解释都把一个角色名映射到解释域的一个二元关系，同时把一个传递性的角色名映射到一个传递性的二元关系。对于所有角色名，逆算子产生的 R^- 被解释为 R 的逆关系。

3.1.1 语法、语义和 SI-表

定义 3.1 设 N_C 是概念名的一个集合；N_R 是全部角色名的一个集合，包括一般性的角色名和传递性的角色名 $N_{R_+} \cup N_{R_p} = N_R$，同时 $N_{R_+} \cap N_{R_p} = \emptyset$；$SI$-角色集合是集合 $N_R \cup \{R^- | R \in N_R\}$。定义 SI-概念集合为满足下列条件的最小集合：

1. 若概念名 $C \in N_C$，则 C 是 SI-概念；

2. 若 C, D 是 SI-概念，R 是 SI-角色，则 $(C \sqcap D)$，$(C \sqcup D)$，$(\neg C)$，$(\forall R.C)$，$(\exists R.C)$ 都是 SI-概念。

定义 3.2 SI-解释 $I=(\Delta^I, \cdot^I)$ 类似定义 2.2，且满足：对任意 $P \in N_R$，$\langle x, y \rangle \in P^I$ 当且仅当 $\langle y, x \rangle \in P^{-I}$ 对任意 $R \in N_{R^+}$，如果 $\langle x, y \rangle \in R^I$ 而且 $\langle y, z \rangle \in R^I$，那么 $\langle x, z \rangle \in R^I$。

概念 C 是可满足的，概念 D 包含概念 C，概念 C, D 等价，以及对于一个解释 I，个体 $x (x \in \Delta^I)$ 是概念 C 的实例，这些定义也都类似定义 2.3。

定义 3.3 为了后面叙述简便，对于角色定义二个函数：

1. 角色的逆算子具有对称性，为了避免 R^{--}，定义函数 Inv 返回角色的逆：

$$Inv(R) := \begin{cases} R^- & \text{如果 } R \in N_R \\ S & \text{否则 } R = S^-, S \in N_R \end{cases}$$

2. 角色 R 是传递的当且仅当 $Inv(R)$ 是传递的。但出现在传递性角色名集合 N_{R_+} 中的可能是 R，也可能是 $Inv(R)$。这样的区分是不必要的，因此定义函数 $Trans$，返回值为真当且仅当 R 是传递性角色，无论 R 是一个角色名还是一个角色名的逆。

$$Trans(R) := \begin{cases} 1 & \text{如果 } R \in N_{R_+} \text{ 或 } Inv(R) \in N_{R_+} \\ 0 & \text{否则} \end{cases}$$

定义 3.4 设 D 是一个 SI-概念（D 是否定范式），R_D 是出现在 D 中的角色以及它们的逆角色组成的集合，即：$R_D = \{R \cup Inv(R) | R \text{ 出现在 } D \text{ 中}\}$。$D$ 的 SI-表 $T=(S, L, \varepsilon)$ 类似定义 2.4，增加性质 6 和性质 7：

(P6) 如果 $\forall R.C \in L(s)$，$\langle s, t \rangle \in \varepsilon(R)$，且 $Trans(R)$，那么 $\forall R.C \in L(t)$

(P7) $\langle x, y \rangle \in \varepsilon(R)$ 当且仅当 $\langle y, x \rangle \in \varepsilon(Inv(R))$

定理 3.1 一个 SI-概念 D 是可满足的当且仅当存在 D 的一个 SI-表 T。

类似定理 2.1，不过由表 T 构造 D 的模型 I 时需要修改对 R 的解释：

$$R^I = \begin{cases} \varepsilon(R)^+ & \text{如果 } Trans(R) \\ \varepsilon(R) & \text{否则} \end{cases}$$

从 R^I 的定义可知，如果 $\langle x, y \rangle \in R^I$，那么：或者 (1) $\langle x, y \rangle \in \varepsilon(R)$；或者 (2) $\langle x, y \rangle \notin \varepsilon(R)$，则存在一条长度大于 1 的路径，使得 $\langle x, s_1 \rangle, \langle s_1, s_2 \rangle, \cdots, \langle s_n, y \rangle \in \varepsilon(R)$。

而由模型 I 构造 D 的表 T 完全同前，表 T 满足性质 1～5。对于性质 6，若不成立，则意味：$x \in (\forall R.C)^I$，$\langle x, y \rangle \in R^I$，$Trans(R)$，但 $y \notin (\forall R.C)^I$ 即存在某个 z 使得 $\langle y, z \rangle \in R^I$，但 $z \notin C^I$。由于角色 R 具有传递性，因此有 $\langle x, z \rangle \in R^I$，而 $z \notin C^I$，这与 $x \in (\forall R.C)^I$ 矛盾，所以 T 满足性质 6。至于性质 7，由逆关系的语义即可得。

3.1.2 SI 的 Tableau-算法

对于 ALC，由于算法扩展规则所增加的新概念都严格小于被分解的概念，因此完整树路径长度 $|sub(D)|$ 步后算法必终止。但传递性角色的加入使得扩展的新概念能与被分解的概念等大，尤其是若 R 是传递的，则 $\forall R.C$ 将沿着 R-标注边原封不动地传播。例如，若结点 x 以 $\{C, \exists R.C, \forall R.(\exists R.C)\}$ 标注，且 $Trans(R)$，则完整树上新增 x 的 R-后续结点 y，y 标注与 x 等同。为了使得扩展过程终止，SI-Tableau-算法引进阻塞 (blocking) 技术，强行中断循环扩展。

SI-Tableau-算法的完整树是在 ALC-完整树的基础上引进以下特点：

如果结点 x 和结点 y 由一条边 $\langle x, y \rangle$ 连接，那么称 y 为 x 的后继，称 x 为 y 的前驱 (predecessor)。先驱 (ancestor) 是前驱关系的传递闭包。由于 ALC 中不允许角色具有传递性，因此未提及先驱定义。

称结点 y 为结点 x 的 R-邻居，如果 y 为 x 的后继，且 $L(\langle x, y \rangle) = R$；或者 y 为 x 的前驱，且 $L(\langle y, x \rangle) = Inv(R)$。由于 ALC 中没有逆角色算子，因此 R-邻居即为 R-后续。

如果结点 x 有先驱 y，使得 $L(x) = L(y)$，则称结点 x 被直接阻塞，结点 y 阻塞了结点 x。如果结点 x 的前驱被阻塞，则称 x 被间接阻塞。总之，x 称为被阻塞，如果存在先驱 y，y 被阻塞或 $L(x) = L(y)$。

SI-Tableau-算法在 ALC 的基础上加入阻塞条件，并增加 $\forall_+$-规则，见表 2。

表 2 SI-Tableau-算法规则

规则名	规则
$\sqcap$-规则	如果 (1) $C_1 \sqcap C_2 \in L(x)$， 而且 x 没有被直接阻塞 (2) $\{C_1, C_2\} \not\subseteq L(x)$ 那么 $L(x) \to L(x) \cup \{C_1, C_2\}$
$\sqcup$-规则	如果 (1) $C_1 \sqcup C_2 \in L(x)$， 而且 x 没有被直接阻塞 (2) $\{C_1, C_2\} \cap L(x) = \emptyset$ 那么 $L(x) \to L(x) \cup \{C\}$， 对于某个 $C \in \{C_1, C_2\}$
$\exists$-规则	如果 (1) $\exists S.C \in L(x)$， 而且 x 没有被阻塞 (1) x 没有一个 S-邻居 y， 使得 $C \in L(y)$ 那么 新增一个结点 y，赋值 $L(\langle x, y \rangle) = S$ 且 $L(y) = \{C\}$
$\forall$-规则	如果 (1) $\forall S.C \in L(x)$， 而且 x 没有被直接阻塞 (2) x 有一个 S-邻居 y， 但 $C \notin L(y)$ 那么 $L(y) \to L(y) \cup \{C\}$
$\forall_+$-规则	如果 (1) $\forall S.C \in L(x)$， 而且 x 没有被直接阻塞 (2) x 有一个 S-邻居 y， 但 $\forall S.C \notin L(y)$ 那么 $L(y) \to L(y) \cup \{\forall S.C\}$

举例说明，设待证可满足性的概念 $\exists S.C \sqcap \forall S.(\neg C \sqcup$

$\neg D)\sqcap\exists R.C\sqcap\forall R.(\exists R.C)$，且角色 R 具有传递性。(1)初始化完整树 T，由$\sqcap$-规则得到树根 ω 的结点标注为 $L(\omega)=\{\exists S.C,\forall S.(\neg C\sqcup\neg D),\exists R.C,\forall R.(\exists R.C)\}$。(2)对 ω 运用$\exists$ -规则分别生成 w 的 S-后续 x 和 R-后续 y，且有 $L(x)=\{C\}$和 $L(y)=\{C\}$。(3)这时再对 w 运用$\forall$ -规则扩充 x 和 y 的结点标注分别得到 $L(x)=\{C,(\neg C\sqcup\neg D)$ 和 $L(y)=\{C,\exists R.C\}$；同时对 ω 还能运用$\forall_+$-规则扩充 y 的结点标注得 $L(y)=\{C,\exists R.C,\forall R.(\exists R.C)\}$。(4)对结点 x 运用$\sqcup$-规则得 $L(x)=\{C,(\neg C\sqcup\neg D),\neg D\}$。(5)对结点 y 运用$\exists$ -规则生成 y 的 R-后续 z，有 $L(z)=\{C\}$，再由$\forall$ -规则和$\forall_+$-规则扩充 z 的结点标注为 $L(z)=\{C,\exists R.C,\forall R.(\exists R.C)\}$。

此时出现 $L(y)=L(z)=\{C,\exists R.C,\forall R.(\exists R.C)\}$，则结点 y 将阻塞结点 z，而被阻塞的结点 z 不能再运用任何算法规则进行扩展。由于已没法继续扩展，因此得到一棵完全的完整树(如图 2 所示)，且 ω,x,y,z 四个结点标注中都没有包含冲突，因而概念 D 是可满足的。

由$\sqcup$-规则的不确定性，在上述步骤(4)中对结点 x 运用$\sqcup$-规则时也可能得到 $L(x)=\{C,(\neg C\sqcup\neg D),\neg C\}$。虽然这是一个冲突，但概念的可满足性是指存在一个解释使得 $D^I\neq\emptyset$，因此这个冲突并不影响上述结点 ω 作为概念 D 的解释的实例。

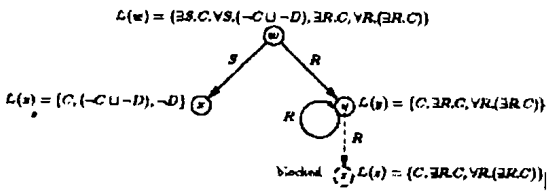


图 2　SI-举例

定理 3.2[界限性]　对任一 SI-概念 D，Tableau-算法都停机。

类似定理 2.2，但由于$\forall_+$-规则的引进，使得完整树路径上的结点标注能够等大传播，而非递减式，因此树的深度不再是原来的 $m=|sub(D)|$。不过，任何结点都是由 $sub(D)$的非空子集标注，所以完整树的路径上最多有 2^m个不同结点，否则就会存在某两个结点有 $L(x)=L(y)$，产生阻塞。因此，算法规则的特征仍决定着 Tableau-算法在有限次使用算法规则后必停机。

定理 3.3[可靠性]　如果对 SI-概念 D 应用算法规则得到一棵完全的、无冲突的完整树，那么 D 存在 SI-表。

类似定理 2.3，利用 Tableau-算法产生的树 T 构造表 $T=(S,L,\varepsilon)$，使其恰为概念 D 的 SI-表。但由于引进了阻塞技术，因此在 S 中要强调是没有被阻塞的结点；同时逆角色算子涉及到前驱结点，因此 $\varepsilon(R)$ 如下定义：

$S=\{x|x$ 是树 T 上的结点，且 x 没有被阻塞$\}$

$\varepsilon(R)=\{(x,y)\in S\times S|1.\ y$ 是 x 的 R-邻居

$2.\ L((x,z))=R$ 且 y 阻塞了 z

$3.\ L((y,z))=Inv(R)$且 x 阻塞了 $z\}$

完整树初始化时 $D\in L(x_0)$，x_0 是根结点，所以不可能被阻塞，$x_0\in S$。即存在个体 $s\in S$ 使得 $D\in L(s)$。由树 T 完全性及无冲突性可证表 T 满足 SI-表的七个性质。

定理 3.4[完备性]　如果 SI-概念 D 存在 SI-表，那么根据 Tableau-算法而恰当运用算法规则，则可产生 D 的一棵完全的、无冲突的完整树。

完全类似定理 2.4，根据概念 D 的 SI-表 $T=(S,L,\varepsilon)$，从初始化树 T 的根结点 x_0，$D\in L(x_0)$起，恰当应用算法规则将产生一棵完全的且无冲突的完整树 T。

定理 3.5[判定性]　对于 SI-概念的可满足性和包含关系，Tableau-算法是可判定的。

定理 3.6[复杂度]　SI-概念的可满足性的判定问题是 PSPACE-完全的。

在 ALC 属于 PSPACE-完全类的证明中，采用的基本搜索技术是：任一时刻只记录唯一一条路径，但由于 SI 中存在逆角色算子，这种方法将不再适用。因为完整树上标注为逆角色的边将导致树的向上扩展，进而对已访问过的上方结点造成影响。如果仍只存储现有的一条路径，而需要扩展的上方结点却早已被存储器删除，那么算法失效。但有效利用阻塞技术仍可建立多项式空间算法[10]。

3.2　SHI

在 SI 的基础上允许存在角色包含公理而形成 SHI 形式系统。在这些公理中出现的角色既可以是传递性的也可以无传递性，同时也可以是角色名或角色名的逆角色。例如，加入两条公理：$R\sqsubseteq R^-$和 $R^-\sqsubseteq R$，则可表示角色 R 具有对称性。

定义 3.5　SHI-概念的定义同定义 3.1。外加：

形如"$R\sqsubseteq S$"称为角色包含公理，其中 R,S 属于 SHI-角色集合。设 R 是由角色包含公理组成的一个集合，则称 R^+为一个角色分层(role hierarchy)，$R^+:=(R\cup\{Inv(R)\sqsubseteq Inv(S)|R\sqsubseteq S\in R\},\dot{\sqsubseteq})$，其中$\dot{\sqsubseteq}$是$\sqsubseteq$在 $R\cup\{Inv(R)\sqsubseteq Inv(S)|R\sqsubseteq S\in R\}$上的自反传递闭包。

定义 3.6　SHI-解释 $I=(\Delta^I,\cdot^I)$在定义 3.2 基础上，还需满足：对任意角色 R,S，$R\dot{\sqsubseteq}S$，如果$\langle x,y\rangle\in R^I$ 那么$\langle x,y\rangle\in S^I$。称解释 I 是 R^+的一个模型，当且仅当对所有 $R\sqsubseteq S\in R^+$都成立 $R^I\subseteq S^I$。

定义 3.7　SHI-表 $T=(S,L,\varepsilon)$同定义 3.4，并修改性质 6，增加性质 8：

(P6')如果$\forall\ S.C\in L(s)$，$\langle s,t\rangle\in\varepsilon(R)$，其中 $Trans(R)$且 $\dot{\sqsubseteq}S$，那么$\forall\ R.C\in L(t)$

(P8)如果$\langle x,y\rangle\in\varepsilon(R)$且 $R\dot{\sqsubseteq}S$，那么$\langle x,y\rangle\in\varepsilon(S)$

SHI-Tableau-算法中将$\forall_+$-规则修改为$\forall'_+$-规则，同时扩充完整树中的 R-邻居的定义。

称结点 y 为结点 x 的 R-邻居，如果 y 为 x 的后继，且 $L((x,y))=S$；或者 y 为 x 的前驱，且 $L((y,x))=Inv(S)$，其中 $S\dot{\sqsubseteq}R$。

表 3　SHI-Tableau-算法规则

规则名	规则
$\forall'_+$-规则	如果　(1)$\forall\ S.C\in L(x)$，而且 x 没有被直接阻塞 (2)存在一个传递性角色 R，且 $R\dot{\sqsubseteq}S$ (3)x 有一个 R-邻居 y，但$\forall\ R.C\notin L(y)$ 那么　$L(y)\to L(y)\cup\{\forall\ R.C\}$

定理 3.7　一个 SHI-概念 D 是可满足的当且仅当存在 D 的一个 SHI-表 T。

类似定理 3.1，但由表 T 构造 D 的模型 I 时如下修改对 R 的解释：

$$R^I=\begin{cases}\varepsilon(R)^+ & \text{如果 } Trans(R)\\ \varepsilon(R)\cup\bigcup\limits_{P\dot{\sqsubseteq}R,P\neq R}P^I & \text{否则}\end{cases}$$

对于非传递性角色的解释是递归定义的，因为这些角色可能

包含传递性的子角色。

从对 R^I 的定义和 *SHI*-表的性质 8 可知，如果$\langle x,y\rangle \in R^I$，那么：或者(1)$\langle x,y\rangle \in \epsilon(R)$；或者(2)$\langle x,y\rangle \notin \epsilon(R)$，则存在一条长度大于 1 的路径，使得$\langle x,s_1\rangle,\langle s_1,s_2\rangle,\cdots,\langle s_n,y\rangle \in \epsilon(S)$，其中 $Trans(S)$ 且 $S \dot{\sqsubseteq} R$。

而由模型 I 构造 D 的表 T 完全同前。表 T 满足性质 1～5 及 7。对于性质 6'，若不成立，即意味：$x \in (\forall S.C)^I$，$\langle x,y\rangle \in R^I$，其中 $Trans(R)$，$R \dot{\sqsubseteq} S$，但 $y \notin (\forall R.C)^I$ 即存在某个 z 使得$\langle y,z\rangle \in R^I$，但 $z \notin C^I$。由角色 R 具有传递性，有$\langle x,z\rangle \in R^I$，再由 $R \dot{\sqsubseteq} S$，得$\langle x,z\rangle \in S^I$，但 $z \notin C^I$，这与 $x \in (\forall S.C)^I$ 矛盾。所以 T 满足性质 6'。至于性质 8，由角色包含关系的语义即可得。

定理 3.8[界限性] 对任一 *SHI*-概念 D，Tableau-算法都停机。

定理 3.9[可靠性] 如果对 *SHI*-概念 D 应用算法规则得到一棵完全的、无冲突的完整树，那么 D 存在 *SHI*-表。

类似定理 3.3 由完整树 T 构造表 T，但需对 ϵ 稍做修改使之满足 *SHI*-表性质。

$\epsilon(R)=\{\langle x,y\rangle \in S\times S \mid$ 1. y 是 x 的 R-邻居，或者

2. 存在角色 S，$S \dot{\sqsubseteq} R$，并且

a. $L(\langle x,z\rangle)=S$ 且 y 阻塞了 z

b. $L(\langle y,z\rangle)=Inv(S)$且 x 阻塞了 $z\}$

定理 3.10[完备性] 如果 *SHI*-概念 D 存在 SHI-表，那么根据 Tableau-算法而恰当运用算法规则，则可产生 D 的一棵完全的、无冲突的完整树。

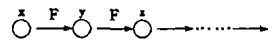
定理 3.11[判定性] 对于 *SHI*-概念的可满足性和包含关系，Tableau-算法是可判定的。

定理 3.12[复杂度] *SHI*-概念的可满足性的判定问题是 EXPTIME-完全的。

事实上，*SI* 和 *SIN* 都是 PSPACE-完全的[10]，但 *SH* 的推理问题就已是 EXPTIME-完全的[8]，所以 *SHI* 以及后面的 *SHIF*，*SHIN*，*SHIQ* 都是 EXPTIME。

3.3 SHIF

数量限定是用于约束角色成员(role filler)的数量的，即允许概念形如$\geqslant nR$ 和$\leqslant nR$。由此在 *SHI* 的基础上扩展而得 *SHIN*，但也因此丧失了模型有限性[6]。例如，角色 R 具有传递性，且 $F \dot{\sqsubseteq} R$，已知概念$\neg C \sqcap \exists F^-.(C \sqcap \leqslant 1F) \sqcap \forall R^-.(\exists F^-.(C \sqcap \leqslant 1F))$是可满足的，但它的任一模型都有一条无穷的 F^--路径。



$L(x)=\{\neg C,\exists F^-.(C\sqcap\leqslant 1F),\forall R^-.(\exists F^-.(C\sqcap\leqslant 1F))\}$

$L(y)=\{(C\sqcap\leqslant 1F),C,\leqslant 1F,\exists F^-(C\sqcap\leqslant 1F),\forall R^-.(\exists F^-.(C\sqcap\leqslant 1F))\}$

$L(z)=\{(C\sqcap\leqslant 1F),C,\leqslant 1F,\exists F^-(C\sqcap\leqslant 1F),\forall R^-.(\exists F^-(C\sqcap\leqslant 1F))\}$

这条路径如果有环路，则要么(1)返回到第一个结点 x，而出现$\{C,\neg C\}\subseteq L(x)$；要么(2)返回到其他任一结点，而$\leqslant 1F$ 矛盾。所以这将是一条无穷路径，并且简单地按照两个结点标注相同就阻塞的这种技术也不再适用，不然结点 y 阻塞了结点 z，而与事实不符。不过，修改算法并引进成对匹配阻塞(pair-wise blocking)技术可以使得产生的仍是有限完整树，但能以此解释无穷论域。

当数量限定$\leqslant nR$ 中限定 $n=1$ 时，称这样描述的概念具有函数性(因为角色成员最多一个，类似自变量最多一个函数值)。函数性约束就是指形如：$\leqslant 1R$ 的概念及其否定式，写成否定范式即$\neg(\leqslant 1R)=\geqslant 2R$。在 *SHI* 中只允许加入函数性约束($\leqslant 1R$ 和$\geqslant 2R$) 形成 *SHIF*[7]，这是 *SHIN* 的特例，也是我们即将讨论的对象。

无论 *SHIF*，*SHIN* 还是 *SHIQ*(下节 4.3 内容)，都要求在约束中出现的角色是简单角色(既不具有传递性，也没有传递的子角色)，否则在算法构造中可能导致后续的结点串全压缩成一个结点。如现有($\leqslant 1S$)，Trans(R)；$R \dot{\sqsubseteq} S$，因为要求满足($\leqslant 1S$)，只好把所有 R-后续形成的结点串挤压成一个结点。更为重要的是，若数量限定中不规定角色为简单的，而放任其随意，则将导致不可判定性[6]。

定义 3.8 *SHIF*-概念在定义 3.5 基础上加入：

若 R 是一个简单角色，则($\leqslant 1R$)，($\geqslant 2R$)是 *SHIF*-概念。

称 R 是一个简单角色，如果$\neg Trans(R)$ 而且对任意 S；$S \dot{\sqsubseteq} R$，S 也是一个简单角色。

定义 3.9 *SHIF*-解释 I 在定义 3.6 基础上加入：

$(\leqslant 1R)^I=\{x\in\Delta^I \mid$对于所有的 y,z，如果$\langle x,y\rangle \in R^I$ 且$\langle x,z\rangle \in R^I$，那么 $y=z\}$

$(\geqslant 2R)^I=\{x\in\Delta^I \mid$存在某两个 y,z，满足$\langle x,y\rangle \in R^I$ 且$\langle x,z\rangle \in R^I$，并有 $y\neq z\}$。

定义 3.10 *SHIF*-表 T 在定义 3.7 基础上加入：

(P9)如果$(\leqslant 1R)\in L(s)$，而且$\langle s,t\rangle \in \epsilon(R)$，$\langle s,t'\rangle \in \epsilon(R)$，那么 $t=t'$

(P10)如果$(\geqslant 2R)\in L(s)$，那么存在 $t,t'\in S$ 满足$\langle s,t\rangle \in \epsilon(R)$，$\langle s,t'\rangle \in \epsilon(R)$且 $t\neq t'$

SHIF-完整树定义大致如前 *SHI*，但由于数量限定使得要对一些结点进行合并，此时边的标注不再只是单角色，而应是一组角色的集合。

称结点 y 为结点 x 的 R-后续，如果 y 为 x 的后继，且 $S \in L(\langle x,y\rangle)$，其中 $S \dot{\sqsubseteq} R$(区别：前述定义中 $S=L(\langle x,y\rangle)$)。称结点 y 为结点 x 的 R-邻居，如果 y 为 x 的 R-后续，或者 x 为 y 的 $Inv(R)$-后续。

采用成对匹配阻塞技术，使得有限完整树可对应无穷论域模型，阻塞定义如下：

(1)如果结点 x 没有先驱被阻塞，而且存在 x 的先驱 x'，y,y'，满足：

1. x 是 x'的后续，y 是 y'的后续

2. $L(x)=L(y)$且 $L(x')=L(y')$

3. $L(\langle x',x\rangle)=L(\langle y',y\rangle)$

则称结点 x 被直接阻塞，称结点 y 阻塞结点 x。

(2)如果结点 x 的前驱被阻塞，或者结点 x 是结点 y 的后续，而 $L(\langle y,x\rangle)=\emptyset$，则称结点 x 被间接阻塞(在 *SHIF* 的 Tableau-算法中边标注可能置空，见表 4 中$\leqslant$-规则)。

完整树中结点 x，称 $L(x)$包含一个冲突，如果：

(1)存在概念 C，有$\{C,\neg C\}\subseteq L(x)$；或者

(2)存在角色 S 和 R，$S \dot{\sqsubseteq}$ 有$\{(\leqslant 1R),(\geqslant 2S)\}\subseteq L(x)$

SHIF-Tableau-算法规则(见表 4)增加两条：$\geqslant$-规则和$\leqslant$-规则，其中 A 是在待证概念 D 中从未出现过的新概念名，这样避免与其他限制彼此干扰。同时由于边的标注扩为集合，还需修改$\exists$-规则(区别：现 $L(\langle x,y\rangle)=\{S\}$与前述规则中 $L(\langle x,y\rangle)=S$)。

定理 3.13 *SHIF*-Tableau-算法具有限定性、可靠性和完全性[7]，计算复杂度为 EXPTIME[8]。

举例说明，设待证可满足性的概念：$\neg C \sqcap \leqslant 1F \sqcap \exists F^{-}.D \sqcap \forall R^{-}.(\exists F^{-}.D)$，且角色 $F \dot{\sqsubseteq} R$，概念 $D=C \sqcap \leqslant 1F \sqcap \exists F.\neg C$。(1)初始化完整树 T，由$\sqcap$-规则得到树根 ω 的结点标注为 $L(w)=\{\neg C, \leqslant 1F, \exists F^{-}.D, \forall R^{-}.(\exists F^{-}.D)\}$。(2)对 w 运用$\exists$-规则生成 w 的 F^{-}-后续 z，有 $L(z)=\{D\}$，再对 w 运用$\forall$-规则和$\forall_{+}$-规则扩充 z 的结点标注为 $L(z)=\{D, \exists F^{-}.D, \forall R^{-}.(\exists F^{-}.D)\}$。(3)由概念 D 的表达式对 z 还能运用$\sqcap$-规则扩充 z 的结点标注得 $L(z)=\{D, \exists F^{-}.D, \forall R^{-}.(\exists F^{-}.D), C, \leqslant 1F, \exists F.\neg C\}$。(4)对结点 z 运用$\exists$-规则生成 z 的 F^{-}-后续 x，同时对 z 再运用$\forall$-规则和$\forall_{+}$-规则，而对 x 运用$\sqcap$-规则扩充 x 的结点标注得 $L(x)=\{D, \exists F^{-}.D, \forall R^{-}.(\exists F^{-}.D), C, \leqslant 1F, \exists F.\neg C\}$。(5)对结点 x 运用$\exists$-规则生成 x 的 F-后续 y，有 $L(y)=\{\neg C\}$。(6)此时如图 3 所示，结点 x 已满足$\leqslant$-规则的前提条件，所以激活该规则合并 x 的这两个 F-邻居 z 和 y 的结点标注得到 $L(z) \leftarrow L(z) \cup L(y)=\{D, \exists F^{-}.D, \forall R^{-}.(\exists F^{-}.D), C, \leqslant 1F, \exists F.\neg C, \neg C\}$。

表 4 *SHIF*-Tableau-算法规则

规则名	规则
$\exists$-规则	如果 (1)$\exists S.C \in L(x)$，而且 x 没有被阻塞 (2)x 没有一个 S-邻居 y，使得 $C \in L(y)$ 那么 新增一个结点 y，赋值 $L(\langle x,y\rangle)=\{S\}$ 且 $L(y)=\{C\}$
$\geqslant$-规则	如果 (1)$\geqslant 2S \in L(x)$，而且 x 没有被阻塞 (2) x 没有 S-邻居 y 使得 $A \in L(y)$ 那么 新增两个结点 y_1, y_2，赋值 $L(y_1)=\{A\}, L(y_2)=\{\neg A\}$ 且 $L(\langle x,y_1\rangle)=\{S\}, L(\langle x,y_2\rangle)=\{S\}$
$\leqslant$-规则	如果 (1)$\leqslant 1S \in L(x)$，而且 x 没有被直接阻塞 (2)x 有两个 S-邻居 y 和 z，且 y 不是 z 的先驱 那么 (1) $L(z) \rightarrow L(z) \cup L(y)$ (2)如果 z 是 y 的先驱，那么 $L(\langle z,x\rangle) \rightarrow L(\langle z,x\rangle) \cup Inv(L(\langle x,y\rangle))$ 否则 $L(\langle x,z\rangle) \rightarrow L(\langle x,z\rangle) \cup L(\langle x,y\rangle)$ (3)$L(\langle x,y\rangle) \rightarrow \emptyset$

于是出现冲突 $\{C, \neg C\} \subseteq L(z)$。事实上，对结点 z 运用$\exists$-规则还将生成 z 的 F-后续 u，有 $L(u)=\{\neg C\}$，那么$\leqslant$-规则也将合并 z 的这两个 F-邻居 w 和 u 的结点标注，不过由于 $L(u) \subseteq L(w)$，因此不会产生影响。当然对结点 x 还有类似步骤(4)的操作，但同样会出现步骤(5)和(6)而产生冲突。综上所述，原概念是不可满足的。

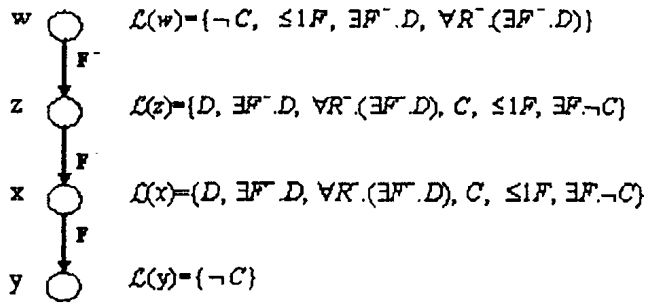


图 3 *SHIF*-举例

3.4 SHIQ

定性数量限定是在数量限定的基础上，进一步限定了角色成员的所属概念，形如：$\geqslant nR.C$ 和 $\leqslant nR.C$。在 *SHIQ*[6,7]中用 $clos(D)$ 代替以前出现的 $sub(D)$，$clos(D)$ 中不仅包括概念 D，且要对其子概念和否定范式封闭。

定义 3.11 *SHIQ*-概念修改定义 3.8 如下：若 R 是一个简单的 *SHIQ*-角色，C 是一个 *SHIQ*-概念，$n \in \mathbb{N}$，则 $(\geqslant nR.C)$，$(\leqslant nR.C)$也是 *SHIQ*-概念。

定义 3.12 *SHIQ*-解释 I 修改定义 3.9 如下：(#表示集合的个数)

$$(\geqslant nR.C)^{I}=\{x \in \Delta^{I} \mid \#(y \in \Delta^{I} \mid \langle x,y\rangle \in R^{I} \text{ 且 } y \in C^{I}\} \geqslant n\}$$

$$(\leqslant nR.C)^{I}=\{x \in \Delta^{I} \mid \#\{y \in \Delta^{I} \mid \langle x,y\rangle \in R^{I} \text{ 且 } y \in C^{I}\} \leqslant n\}$$

定义 3.13 *SHIQ*-表 T 中 $clos(D)$ 代替 $sub(D)$，并在定义 3.7 基础上加入：

(P 9) 如果$(\geqslant nR.C) \in L(s)$，那么 $\#\{t \in S \mid \langle s,t\rangle \in \varepsilon(R), c \in L(t)\} \geqslant n$

(P10) 如果$(\leqslant nR.C) \in L(s)$，那么 $\#\{t \in S \mid \langle s,t\rangle \in \varepsilon(R), C \in L(t)\} \leqslant n$

(P11)如果$(\Diamond nR.C) \in L(s)$，且$\langle s,t\rangle \in \varepsilon(R)$，那么 $C \in L(t)$或者$\sim C \in L(t)$。

其中$\Diamond$是$\geqslant$和$\leqslant$的占位符，$\sim C$是$\neg C$的否定范式。

事实上，对任意解释 I，概念 C 以及个体 x，都有 $x \in C^{I}$ 或者 $x \in (\neg C)^{I}$，而在前述定义的各种形式语言的表中都没有对此显式表达。但对于 *SHIQ*，为了保证性质(9)和性质(10)的正确性，必须新增性质(11)，借此强行将 C 或$\neg C$ 加入相关结点的标注中，以便累计集合元素的个数。

SHIQ-完整树 T 上的每个结点 x 都被概念组成的一个集合标注，$L(x) \subseteq clos(D)$；每条边$\langle x,y\rangle$都被角色组成的一个集合标注，$R \in L(\langle x,y\rangle)$；而树上不同的结点由二元关系$\neq$划分。

完整树中结点 x，称 $L(x)$包含一个冲突，如果：

(1)存在概念 C，有$\{C, \neg C\} \subseteq L(x)$；或者

(2)存在概念 C，角色 R 和自然数 $n \in \mathbb{N}$，有$(\leqslant nR.C) \in L(x)$，却又有 $n+1$ 个结点 $y_0, \cdots, y_n$ 作为 x 的 R-邻居，满足 $C \in L(y_i)$且 $y_i \neq y_j$ 对所有 $0 \leqslant i < j \leqslant n$ 成立。

初始化完整树 T 仅包含一个结点 x_0(根结点)，$L(x_0)=\{D\}$，D 为待证可满足性的概念，不等关系$\neq$初始化为空集。

SHIQ-Tableau-算法规则(见表 5)修改 *SHIF* 中$\geqslant$-规则和$\leqslant$-规则。同时根据性质(11) 新增“选择”-规则。

定理 3.14 *SHIQ*-Tableau-算法具有限定性、可靠性和完全性[7]，复杂度为 EXPTIME[8]。

现已实现的 FaCT (Fast Classification of Terminolo-

gies)系统[21]可以对 *SHIQ*-语言进行知识表示和推理；而 RACE(Reasoner for ABoxes and Concept Expression)推理机最初用于 *SHN* 形式系统，在此基础上研发的 RACER(Renamed ABox and Concept Expression Reasoner)[22]，则同样支持 *SHIQ*。在这些切实可行的 EXPTIME-推理机中，优化后的 Tableau-算法具有举足轻重的地位。

表 5 *SHIQ*-Tableau-算法规则

规则名	规则
选择-规则	如果 (1)$\Diamond nS.C\in L(x)$，而且 x 没有被直接阻塞 (2)y 是 x 的 S-邻居，但$\{C,\sim C\}\cap L(y)=\emptyset$ 那么 $L(y)\rightarrow L(y)\cup\{E\}$，对于某个 $E\in\{C,\sim C\}$
≥-规则	如果 (1)$\geqslant nS.C\in L(x)$，而且 x 没有被阻塞 (2)x 没有 n 个 S-邻居 $y_1,\cdots,y_n$，使得 $C\in L(y_i)$且 $y_i\neq y_j$ 对所有 $1\leqslant i\leqslant j\leqslant n$ 成立 那么 新增 n 个结点 $y_1,\cdots,y_n$，赋值 $L(\langle x,y_i\rangle)=\{S\}$，且 $L(y_i)=\{C\}$，其中 $y_i\neq y_j,1\leqslant i<j\leqslant n$
≤-规则	如果 (1)$\leqslant nS.C\in L(x)$，而且 x 没有被直接阻塞 (2)$\#\{u\mid u$ 是 x 的 S-邻居，且 $C\in L(u)\}>n$， 存在 x 的两个 S-邻居 y 和 z 满足 $C\in L(y),C\in L(z)$，且 y 不是 z 的先驱，也没出现 $y\neq z$ 那么 (1)$L(z)\rightarrow L(z)\cup L(y)$ (2) 如果 z 是 y 的先驱，那么 $L(\langle z,x\rangle)\rightarrow L(\langle z,x\rangle)\cup Inv(L(\langle x,y\rangle))$ 否则 $L(\langle x,z\rangle)\rightarrow L(\langle x,z\rangle)\cup L(\langle x,y\rangle)$ (3)$L(\langle x,y\rangle)\rightarrow\emptyset$ (4)对所有 u，若有 $u\neq y$，则 $u\neq z$

4 SHOQ(D)

语义 Web 的目标是充分利用网络资源(不仅限于 HTML 网页，包括所有可获取的数据及服务)，实现计算机自动存取信息。机器可读可理解可处理，自然要求资源的描述具有语义，可借助元数据及本体建立语义 Web 体系结构[14,15]。OWL 是由 W3C 网络本体工作组最新推出的语义 Web 本体语言，现已成为 W3C 的标准[19]。

描述逻辑不仅为 OWL 提供了基础理论，并有助于解决实际推理问题。一般而言，OWL Lite 的本体推理可归约为 *SHIF*(*D*) 的知识库可满足性，Tableau-算法对后者可判定且计算复杂度为 EXPTIME，而 OWL DL 则可归约到 *SHOIN*(*D*)，后者的推理具有 NEXPTIME 计算复杂度[16~18]。FaCT 和 RACER 都是现已实现的描述逻辑系统，可为 OWL Lite 提供推理服务，但对于 *SHOIN*(*D*) 尚无实用算法，因此 OWL DL的推理实现还需深入探讨。

SHIF 和 *SHIQ* 已在 3.3 和 3.4 节中介绍，*O* 和 *D* 将在 4.1 和 4.2 节中引进。关于逆角色算子的推理是比较棘手的，即便在最基本的 *ALC* 上同时允许逆角色(*I*)、有型域(*D*)和枚举算子(*O*)，算法复杂度也会由 PSPACE 跳跃至 NEXPTIME[13]。另外，个体的描述将使任何 DLs 都丧失树模型属性，若再有逆角色、数量限定等，则问题更加复杂[12]。因此 4.3 节选择 SHOQ(D)作为对语义 Web 中描述逻辑的 Tableau-算法的介绍。事实上在 OWL 提出之前推行的语义 Web 上，构建当时的本体语言 OIL 时就已考虑到了将 OIL 的框架语法映射为 SHOQ(D)的公理[18]。

4.1 O

现实生活中关于个体的描述是常见的，有时通过枚举实例来描述一个概念，如联合国常任理事国＝{中国，法国，俄罗斯，英国，美国}，有时用单元素集合(singleton)对个体进行推理，如查找“国籍为中国的人”，“中国”就是命名个体(named individual)视为单元素集合。描述逻辑中集合算子(set) 也称枚举算子(oneOf)，可将个体名整合成一个集合概念[1]，简称其为 *O*。

由个体名 $a_1,\cdots,a_n$ 构建集合概念$\{a_1,\cdots,a_n\}$，解释为：$\{a_1,\cdots,a_n\}^I=\{a_1^I,\cdots,a_n^I\}$

事实上，单元素集合就足以描述任意有限集合，因为概念$\{a_1,\cdots,a_n\}$与概念$\{a_1\}\sqcup\cdots\sqcup\{a_n\}$等价。

由此可知，“国籍为中国的人”将被描述为新概念：

人⊓(∃ 国籍.{中国})。

4.2 D

有型域(concrete domain，简称其为 D)是对描述逻辑的一个扩展，使之包括数值、字符串、时间等这类有型对象[12]，在语义 Web 中对应着数据类型(datatype)[19]。下面就最基本的 *ALC* 上扩展得 *ALC*(*D*)。

在 DLs 的有型域扩展中，抽象个体(抽象域 Δ^I中的元素)通过特征式(feature，即函数性角色)映射为有型域上的值(如整数，字符串等)，这些具体数据再被有型域上的谓词公式约束。例如，($Person\sqcap\exists$ ($income$; $expenses$).$\leqslant$) 这个概念描述的是花钱比赚钱多的那些人，*income*，*expenses* 是特征式，≤是有型域(整数)上的谓词公式。

定义 4.1 有型域 D 是一个二元组$(\Delta^D,pred(D))$，其中 Δ^D 是有型论域，$pred(D)$ 是谓词集合。任意 n 元谓词 $P\in pred(D)$ 是论域上的 n 元关系，即：$P^D\subseteq(\Delta^D)^n$。

定义 4.2 设 N_C为概念名集合，N_R为角色名集合，N_f为特征式集合，且彼此不交。*ALC*(*D*)-角色集合是 $N_R\cup N_f$。特征链是指特征式的合成 $f_1\circ f_2\circ\cdots\circ f_k$。

定义 *ALC*(*D*)-概念集合为满足下列条件的最小集合：

1. 若概念名 $C\in N_C$，则 C 是 *ALC*(*D*)-概念。

2. 若 C,D 是 *ALC*(*D*)-概念，R 是 *ALC*(*D*)-角色，则(C⊓D)，(C⊔D)，(¬C)，(∀R.C)，(∃R.C)都是 *ALC*(*D*)-概念。

3. 若 $u_1,\cdots,u_n$ 是特征链，$P\in pred(D)$是 n 元谓词，则∃$u1\cdots u_n.P$ 是 *ALC*(*D*)-概念。

定义 4.3 *ALC*(*D*)-解释 $I=(\Delta^I,\cdot^I)$ 由非空集合 Δ^I(抽象论域)和函数 $\cdot^I$组成。集合 Δ^I与 Δ^D 不交。函数 $\cdot^I$将每个概念映射为 Δ^I 的一个子集，将每个角色映射为 $\Delta^I\times\Delta^I$ 的一个子集，同时还将每个特征式映射为从 $\Delta^I\times\Delta^D$ 的一个子

集。即：$f^I(x)=a$，其中 f 是特征式，抽象个体 $x\in\Delta^I$，函数值 $a\in\Delta^D$，记做$\langle x,a\rangle\in f^I$。特征链 $u=f_1\circ\cdots\circ f_k$的解释为函数的合成 $u^I=f_1^I\circ\cdots\circ f_k^I$。

在$(C\sqcap D)^I,(C\sqcup D)^I,(\neg C)^I,(\forall R.C)^I,(\exists R.C)^I$原有语义基础上，新增：

$(\exists u_1\cdots u_n.P)^I=\{x\in\Delta^I\mid$存在 $a_1,\cdots,a_n\in\Delta^D$，满足$\langle x,a_1\rangle\in u_1^I\wedge\cdots\wedge\langle x,a_n\rangle\in u_n^I,(a_1,\cdots,a_n)\in P^D\}$

Carsten Lutz 已证：如果有型域 D 上的推理问题是 PSPACE，那么 $ALC(D)$ 的判定问题依旧是 PSPACE-完全的[12]。但实际应用中，$ALC(D)$ 的描述能力还不足，尤其是在语义 Web 的范畴下，更倾向于 $SHIF(D)$，$SHOIN(D)$ 等实用形式语言系统。事实上，推理机 RACE 和 RACER，都已在原有基础上纳入有型域，但只允许特征式，不允许特征链，且这样的扩展未影响算法的可判定性和计算复杂度[22]。

4.3 SHOQ(D)

首先，$SHOQ(D)$ 扩展定义集合 $cl(D)$：不仅包括概念 D 的所有子概念，这些子概念的否定范式，还包括这些子概念中出现的数据类型或其否定式。角色包含公理、角色分层，以及简单角色的定义如前，但无“逆角色”构子。

其次，类似 4.2 节中介绍的有型域 D，但作了部分简化：(1)无需 n 元谓词，仅讨论一元谓词；(2)无需特征链，仅讨论一个有型角色。因此不妨按语义 Web 的惯例，称 D 为数据类型，其主要讨论对象为：存在性数据类型$\exists F.d$ 和任意性数据类型$\forall F.d$。例如，描述概念“小于 25 岁的人”为 $Person\sqcap\exists age.(min25)$，其中 age 是有型角色，其值为代表年龄的整数，$(min25)$是在数据类型上的一元谓词。

定义 4.4 设 N_C 是概念名集合，$N_R=N_R^A\uplus N_R^D$是角色名集合(N_R^A抽象角色名集合，N_R^D是有型角色名集合)，N_I 是个体名集合，D 是数据类型集合。

定义 $SHOQ(D)$-概念为满足下列条件的最小集合：

1. 若概念名 $C\in N_C$，则 C 是一个 $SHOQ(D)$-概念；
2. 若个体名 $o\in N_I$，则 o 是一个 $SHOQ(D)$-概念；
3. 若 C,D 是 $SHOQ(D)$-概念，R 是一个抽象角色，F 是一个有型角色，S 是一个简单角色，$d\in D$ 是一个数据类型，则$(C\sqcup D),(C\sqcap D),(\neg C),(\neg d),(\exists R.C),(\forall R.C),(\geqslant nS.C),(\leqslant nS.C),(\exists F.d),(\forall F.d)$都是 $SHOQ(D)$-概念。

定义 4.5 $SHOQ(D)$-解释 $I=(\Delta^I,\cdot^I)$由非空集合 Δ^I(抽象论域)和函数 $\cdot^I$组成。Δ^D是所有数据类型的论域，集合 Δ^I与 Δ^D不交。函数 $\cdot^I$将每个概念映射为 Δ^I的一个子集(包括个体)，将每个抽象角色映射为 $\Delta^I\times\Delta^I$的一个子集，将每个有型角色映射为 $\Delta^I\times\Delta^D$的一个子集。

$SHOQ(D)$-解释增加如下语义：

个体 $o^I\subseteq\Delta^I;\#o^I=1$

数据类型 $d^D\subseteq\Delta^D$

有型角色 $F^I\subseteq\Delta^I\times\Delta^D$

$(\exists F.d)^I=\{x\in\Delta^I\mid$存在一 $y\in\Delta^D$，满足$\langle x,y\rangle\in F^I$，而且 $y\in d^D\}$

$(\forall F.d)^I=\{x\in\Delta^I\mid$对任意 $y\in\Delta^D$，如果$\langle x,y\rangle\in F^I$，那么 $y\in d^D\}$

定义 4.6 设 D 是一个 $SHOQ(D)$-概念(D 是否定范式)，R^+是一个角色分层，R_D^A和 R_D^D 分别是出现在 D 或者 R^+中的抽象角色和有型角色组成的集合。定义 D 关于 R^+的 $SHOQ(D)$-表 T 为一个四元组$(S,L,\varepsilon_A,\varepsilon_D)$：

S：个体的集合；

$L:S\to 2^{cl(D)}$将 S 中的个体映射为 $cl(D)$ 的子集；

$\varepsilon_A:R_D^A\to 2^{S\times S}$将 R_D^A 中的抽象角色映射为个体-个体对的集合；

$\varepsilon_D:R_D^D\to 2^{S\times\Delta^D}$将 R_D^D 中的有型角色映射为个体-数据类型值的集合。

同时要求存在某个个体 $s\in S$ 使得 $D\in L(s)$。对任意 $s,t\in S,C,C_1,C_2\in cl(D),R,S\in R_D^A,F,F'\in R_D^D$，$SHOQ(D)$-表有下列 13 个性质：

(P1)如果 $C\in L(s)$，那么$\neg C\notin L(s)$

(P2)如果 $C_1\sqcap C_2\in L(s)$，那么 $C_1\in L(s)$而且 $C_2\in L(s)$

(P3)如果 $C_1\sqcup C_2\in L(s)$，那么 $C_1\in L(s)$或者 $C_2\in L(s)$

(P4)如果$\forall R.C\in L(s)$，而且$\langle s,t\rangle\in\varepsilon_A(R)$，那么 $C\in L(t)$

(P5)如果$\exists R.C\in L(s)$，那么存在某个 $t\in S$，使得$\langle s,t\rangle\in\varepsilon_A(R)$，而且 $C\in L(t)$

(P6)如果$\forall S.C\in L(s)$，而且$\langle s,t\rangle\in\varepsilon_A(R),Trans(R),\dot{R\subseteq}S$，那么$\forall R.C\in L(t)$

(P7)如果$\langle s,t\rangle\in\varepsilon_A(R)$且 $\dot{R\subseteq}S$，则$\langle s,t\rangle\in\varepsilon_A(s)$；如果$\langle s,t\rangle\in\varepsilon_D(F)$且 $\dot{F\subseteq}F'$，则$\langle s,t\rangle\in\varepsilon_D(F')$

(P8)如果$(\geqslant nS.C)\in L(s)$，那么$\#\{t\in S\mid\langle s,t\rangle\in\varepsilon_A(S),C\in L(t)\}\geqslant n$

(P9)如果$(\leqslant nS.C)\in L(s)$，那么$\#\{t\in S\mid\langle s,t\rangle\in\varepsilon_A(S),C\in L(t)\}\leqslant n$

(P10)如果$(\diamond nS.C)\in L(s),\langle s,t\rangle\in\varepsilon_A(S)$，那么 $C\in L(t)$或者$\sim C\in L(t)$

(P11)如果 $o\in L(s)\bigcap L(t)$，那么 $s=t$

(P12)如果$\forall F.d\in L(s)$而且$\langle s,t\rangle\in\varepsilon_D(F)$，那么 $t\in d^D$

(P13)如果$\exists F.d\in L(s)$，那么存在某个 $t\in\Delta^D$，使得$\langle s,t\rangle\in\varepsilon_D(F)$，而且 $t\in d^D$

其中$\diamond$是$\geqslant$和$\leqslant$的占位符，$\sim C$ 是$\neg C$ 的否定范式，# 表示集合的个数。表的性质 11 确保任一个体都被解释为独立体，性质 12 和 13 确保数据类型解释的正确性。

定理 4.1 一个 $SHOQ(D)$-概念 D 关于 R^+是可满足的当且仅当存在 D 的一个关于 R^+的 $SHOQ(D)$-表。

设 R^+是一个角色分层，D 是一个 $SHOQ(D)$-概念(D 是否定范式)，R_D^A 是出现在 D 或者 R^+的抽象角色组成的集合，I_D 是出现在 D 中的个体组成的集合。概念 D 关于 R^+的 $SHOQ(D)$-完整森林 F 是一族树 T，森林里的每个结点 x 都被一个集合标注，$L(x)\subseteq cl(D)\cup\{(\uparrow(R,o)\mid R\in R_D^A$ 且 $o\in I_D\}$；森林里的每条边$\langle x,y\rangle$都被角色组成的一个集合标注，$R\in L(\langle x,y\rangle)$($R$ 出现在 $cl(D)$或者 R^+中)；森林里不同的结点由二元关系$\neq$划分。对任一个体 $o\in I_D$，在森林里定义一个鉴证点 x_o(distinguished node)使得 $o\in L(x_o)$。用 $\uparrow(R,o)\in L(x)$表示一条从 x 到 x_o 的以 R 为标注的边。

称结点 y 为结点 x 的 R-后续(x 称为 y 的前驱)，如果 $S\dot{\subseteq}R$，或者(1)y 为 x 的后继，且 $S\in L(\langle x,y\rangle)$；或者(2)$\uparrow(R,o)\in L(x)$且 $y=x_o$。先驱是前驱关系的传递闭包(注：没有 R-邻居的定义，因为没有逆角色算子)。

称结点 x 被直接阻塞，如果结点 x 的所有先驱都没有被阻塞，而且 x 有一个非鉴证点的先驱 y，使得 $L(x)\subseteq L(y)$，对此也称 y 阻塞了 x。称结点 x 被间接阻塞，如果 x 的前驱被阻塞。总之，如果结点被直接阻塞，或它的前驱被阻塞，都称该结点被阻塞(注：因为没有逆角色算子，所以只需借助子集阻塞技术(subset blocking)[9])。

表 6 $SHOQ(D)$-Tableau-算法规则

规则名	规则
⊓-规则	如果 (1) $C_1 \sqcap C_2 \in L(x)$，而且 x 没有被直接阻塞 (2) $\{C_1, C_2\} \not\subseteq L(x)$ 那么 $L(x) \rightarrow L(x) \cup \{C_1, C_2\}$
⊔-规则	如果 (1) $C_1 \sqcup c_2 \in L(x)$，而且 x 没有被直接阻塞 (2) $\{C_1, C_2\} \cap L(x) = \emptyset$ 那么 $L(x) \rightarrow L(x) \cup \{C\}$，对于某个 $C \in \{C_1, C_2\}$
∃-规则	如果 (1)$\exists R.C \in L(x)$，而且 x 没有被阻塞 (2)x 没有一个 R-后续 y，使得 $C \in L(y)$ 那么 新增一个结点 y，赋值 $L(\langle x,y\rangle) = \{R\}$ 且 $L(y) = \{c\}$ 如果 (1)$\exists F.d \in L(x)$，而且 x 没有被阻塞 (2)x 没有一个 F-后续 y，使得 $d \in L(y)$ 那么 新增一个结点 y，赋值 $L(\langle x,y\rangle) = \{F\}$ 且 $L(y) = \{d\}$
∀-规则	如果 (1)$\forall R.C \in L(x)$，而且 x 没有被直接阻塞 (2)x 有一个 R-后续 y，但 $C \notin L(y)$ 那么 $L(y) \rightarrow L(y) \cup \{C\}$ 如果 (1)$\forall F.d \in L(x)$，而且 x 没有被直接阻塞 (2)x 有一个 F-后续 y，但 $d \notin L(y)$ 那么 $L(y) \rightarrow L(y) \cup \{d\}$
∀+-规则	如果 (1)$\forall S.C \in L(x)$，而且 x 没有被直接阻塞 (2)存在一个传递性角色 R，且 $R \dot{\sqsubseteq} S$ (3)x 有一个 R-后续 y，但$\forall R.C \notin L(y)$ 那么 $L(y) \rightarrow L(y) \cup \{\forall R.C\}$
选择-规则	如果 (1)$\Diamond nS.C \in L(x)$，而且 x 没有被直接阻塞 (2)y 是 x 的 S-后续，且$\{C, \sim C\} \cap L(y) = \emptyset$ 那么 $L(y) \rightarrow L(y) \cup \{E\}$，对于某个 $E \in \{C, \sim C\}$
≥-规则	如果 (1)$\geqslant nS.C \in L(x)$，而且 x 没有被阻塞 (2)x 没有 n 个 S-后续 $y_1, \cdots, y_n$，使得 $C \in L(y_i)$且 $y_i \neq y_j$ 对所有 $1 \leqslant i < j \leqslant n$ 都成立 那么 新增 n 个结点 $y_1, \cdots, y_n$，赋值 $L(\langle x,y_i\rangle) = \{S\}$，$L(y_i) = \{C\}$，其中 $y_i \neq y_j$，$1 \leqslant i < j \leqslant n$
≤-规则	如果 (1)$\leqslant nS.C \in L(x)$，而且 x 没有被直接阻塞 (2)x 有 $n+1$ 个 S-后续 $y_0, \cdots, y_n$，满足 $C \in L(y_i)$，$0 \leqslant i \leqslant n$ (3)存在 $i \neq j$ 却没出现 $y_i \neq y_j$，且若 y_i, y_j 中有仅有一个是独行点，那一定是 y_i 那么 (1)$L(y_i) \rightarrow L(y_i) \cup L(y_j)$ (2)如果 y_i, y_j 都不是独特点，那么 $L(\langle x,y_i\rangle) \rightarrow L(\langle x,y_i\rangle) \cup L(\langle x,y_j\rangle)$； 如果 y_i 是而 y_j 不是，那么 $L(x) \rightarrow L(x) \cup \{\uparrow(S,o) \mid S \in L(\langle x,y_i\rangle)\}$，对于某个 $o \in L(y_i)$ (3)除去 y_j 和所有在完整森林里与 y_j 相连的边 (4)令 $u \neq y_i$，对所有 $u, u \neq y_j$ 都成立
O-规则	如果 (1)$o \in L(x)$，而且 x 没有被直接阻塞 (2)x 不是独特点，也未出现 $x \neq x_o$ 那么 对 $o \in L(z)$的独特点 z，赋值： (1)$L(z) \rightarrow L(z) \cup L(x)$ (2)如果 x 有前驱 x'，则 $L(x') \rightarrow L(x') \cup \{\uparrow(R,o) \mid R \in L(\langle x',x\rangle)\}$ (3)除去 x 和所有在完整森林里与 x 相连的边 (4)令 $u \neq z$，对所有 $u, u \neq x$ 都成立。

完整森林 F 里结点 x，称 $L(x)$包含一个冲突，如果：

(1)存在概念 C，有$\{C, \neg C\} \subseteq L(x)$；

(2)存在概念 C，角色 S 和自然数 $n \in \mathbb{N}$，有$(\leqslant nS.C) \in L(x)$，却又有 $n+1$ 个结点 $y_0, \cdots, y_n$ 作为 x 的 S-后续，满足 $C \in L(y_i)$且 $y_i \neq y_j$ 对所有 $0 \leqslant i < j \leqslant n$ 成立；

(3)$L(x)$包含数据类型(或否定式)$d_1, \cdots, d_n$，但 $d_1^D \cap \cdots \cap d_n^D = \emptyset$；

(4)存在某个体 $o \in L(x)$，但 $x \neq x_o$。

待证可满足性的概念 D，$o_1, \cdots, o_n$ 是在 D 中出现的全部个体。初始化完整森林 F 包含 $n+1$ 个根结点 $x_o, x_{o_1}, \cdots, x_{o_n}$，赋值 $L(x_o) = \{D\}$以及 $L(x_{o_i}) = \{o_i\}$，不等关系 $\neq$ 初始化为空集。用 $SHOQ(D)$的 Tableau-算法规则(见表 6)来扩展该完整森林 F 直至完全(或者某个结点的标注 $L(x)$中出现冲突，或者没有规则可再用)。

定理 4.2 $SHIO(D)$-Tableau-算法具有限定性、可靠性和完全性，对概念可满足性和包含关系是可判定的[11]。

结论 语义 Web 是新一代互联网的基础，OWL 是 W3C 最新提出的一种强有力的语义 Web 语言，随其发展，描述逻辑也日益受到关注和重视。除了 DL 基本构子，OWL 根据自身需要还整合了集合概念以及数据类型作为其形式语言的构建成分，而 OWL 本体推理问题，则可转变为 DL 的概念可满足性判定，从而得以在 DL 的推理机上实现。

基于对上述问题的关注，本文从形式化方面依次介绍了描述逻辑语言 ALC，SI，SHI，$SHIF$，$SHIQ$ 以及 $SHOQ(D)$

及其对应的 Tableau 算法。每种形式系统都定义了关于概念的表，证明了概念的可满足性等价于这样的表的存在性。由此可知，概念的可满足性判定问题关键在于能否构造出合法的表。Tableau 算法正是通过构造一棵完全的完整树来实现这一目的。当树的某个结点中存在冲突，则概念不可满足，否则得到的是一棵完全的、无冲突的完整树，即意味着概念是可满足的。Tableau 算法规则的合理设计和一些巧妙的技术手段（如阻塞技术）使得 Tableau 算法具有可靠性完全性，并在有限步内必停机。因此在描述逻辑中，Tableau 算法常常用于判定概念的可满足性问题，进而也能对概念的包含关系、知识库的推理问题等做出判断。

虽然 20 世纪 90 年代初期仍将所有非线性复杂度的算法归为不实用算法，但实践已证明描述逻辑的 Tableau-算法是推理机中的实用技术，尽管最基本的 *ALC* 就已经属于 PSPACE-完全类。Ian Horrocks 基于 EXPTIME-完全类的 *SH* 形式语言而实现了 FaCT 系统，不仅可以表示医学术语，而且随着逐步优化和改善，可用于更广泛的推理。Volker Haarslev，RalfMöller 等人研发的 RACE 和 RACER 是关于 *SHN*，*SHIQ* 知识库的推理机，最新版本已包括有型域的表示和推理。

虽然最新的本体语言 OWL Lite 可用 FaCT 或 RACER 进行推理，但描述能力更强的语义 Web 语言 OWL DL 暂无实用推理机，这必将促进与之相关的 $SHOIN(D)$ 的 Tableau-算法的进一步研究。

此外，同样能够有效进行知识表示和推理的逻辑程序设计已发展得较为成熟，将描述逻辑和逻辑程序设计结合在一起形成描述逻辑程序设计（Description Logic Programs，DLP）也是最近的研究热点[24]。

附录：描述逻辑的语法和语义

构子	语法	语义	简称
概念名	A	$A^I\subseteq\Delta^I$	*ALC*
角色名	R	$R^I\subseteq\Delta^I\times\Delta^I$	
合取	$C\sqcap D$	$C^I\cap D^I$	
析取	$C\sqcup D$	$C^I\cup D^I$	
否定	$\neg C$	Δ^I/C^I	
存在性限定	$\exists\, R.C$	$\{x\mid\exists\, y.\langle x,y\rangle\in R^I\wedge y\in C^I\}$	
值限定	$\forall\, R.C$	$\{x\mid\forall\, y.\langle x,y\rangle\in R^I\rightarrow y\in C^I\}$	
传递性角色	$R\in N_{R+}$	$R^I=(R^I)^+$	R^+
逆角色	R^-	$\{\langle x,y\rangle\mid\langle y,x\rangle\in R^I\}$	I
角色分层	$R\sqsubseteq S$	$R^I\subseteq S^I$	H
数量限定	$\leqslant nR$ $\geqslant nR$	$\{x\mid\#\{y.\langle x,y\rangle\in R^I\}\leqslant n\}$ $\{x\mid\#\{y.\langle x,y\rangle\in R^I\}\geqslant n\}$	N
函数性约束	$\leqslant 1R$ $\geqslant 2R$	$\{x\mid\forall\, y.\forall\, z.\langle x,y\rangle\in R^I\wedge\langle x,z\rangle\in R^I\rightarrow y=z\}$ $\{x\mid\exists\, y.\exists\, z.\langle x,y\rangle\in R^I\wedge\langle x,z\rangle\in R^I\wedge y\neq z\}$	F
定性数量限定	$\leqslant nR.C$ $\geqslant nR.C$	$\{x\mid\#\{y.\langle x,y\rangle\in R^I\wedge y\in C^I\}\leqslant n\}$ $\{x\mid\#\{y.\langle x,y\rangle\in R^I\wedge y\in C^I\}\geqslant n\}$	Q
枚举算子	o	$o^I\subseteq\Delta^I,\#o^I=1$	O
数据类型	d F $\exists\, F.d$ $\forall\, F.d$	$d^D\subseteq\Delta^D$ $F^I\subseteq\Delta^I\times\Delta^D$ $\{x\in\Delta^I\mid\exists\, y.\langle x,y\rangle\in F^I\wedge y\in d^D\}$ $\{x\in\Delta^I\mid\forall\, y.\langle x,y\rangle\in F^I\rightarrow y\in d^D\}$	D

参 考 文 献

1 Baader F, Nutt W. Handbook of Description Logic, the second chapter. Cambridge University Press, Jan. 2003

2 Schmidt-Schauβ M, Smolka G. Attributive concept descriptions with complements. Artificial Intelligence, 1991, 48(1): 1～26

3 Sattler U. A concept language extended with different kinds of transitive roles. In 20. Deutsche Jahrestagung für KI, LNAI 1137. Springer-Verlag, 1996

4 Horrocks I, Gough G. Description logics with transitive roles. In: M. -C. Rousset, R. Brachmann, F. Donini, E. Franconi, I. Horrocks, and A. Levy, eds, Proc. of DL'97, 1997. 25～28

5 Schild K. A correspondence theory for terminological logic: Preliminary report. In: Proc. of IJCAI'91, Sydney, 1991. 466～471

6 Horrocks I, Sattler U, Tobies S. Practical reasoning for expressive description logics. In: Harald Ganzinger, David McAllester and Andrei Voronkov, eds, Proc. of the 6th Int. Conf. on Logic for Programming and Automated Reasoning (LPAR'99), number 1705 in Lecture Notes in Artificial Intelligence, Spinger, 1999. 161～180

7 Horrocks I, Sattler U, Tobies S. A description logic with transitive and converse roles, role hierarchies and qualifying number restrictions: [LTCS-Report 99-08]. LuFg Theoretical Computer Science, RWTH Aachen, Germany, 1999

8 Horrocks I, Sattler U, Tobies S. Practical reasoning for very expressive description logics. Logic Journal of the Interest Group in Pure and Applied Logic, 2000, 8(3): 239～264

9 Baader F, Sattler U. An Overview of Tableau Algo-rithms for Description Logics. Studia Logica, 2001, 69: 5～40

10 Horrocks I, Sattler U, Tobies S. A PSPACE-algorithm for deciding $ALCNI_{R_+}$-satisfiability: [LTCS-Report 98-08]. LuFg Theoretical Computer Science, RWTH Aachen, Germany, 1998

11 Horrocks I, Sattler U. Ontology reasoning in the $SHOQ(D)$ description logic. In: Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI'2001), 2001. 199～204

12 Lutz C. The complexity of Reasoning with Concrete Domains: [Ph. D thesis]. LuFG Theoretical Computer Science, RWTH Aachen, Germany, 2002

（下转第 35 页）

在给定的保证QoS的界限以下，图6显示BMDCA的CBP比GC高，而比MDCA低。图7显示GC的信道利用率最高，而它不能保证QoS。BMDCA的信道利用率比MDCA高，说明控制精度有提高。

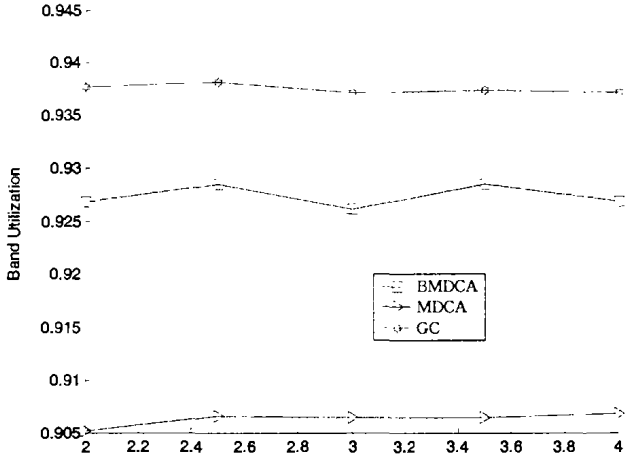


图7 总的带宽利用率的比较

结论 本文给出了一种有效的多业务动态呼叫接纳控制策略，用以支持无线多媒体呼叫切换的QoS。在保证CDP小于QoS界的前提下，使无线网络的信道利用率尽可能高。该策略具有如下独特的优点：1)能为多业务提供精确的服务保证和服务区分；2)由于其动态特征，能动态地适应系统参数和流量的变化；3)由于模型考虑了带宽的限制、呼叫中断概率(call dropping probability)的时变特征，以及相邻或相近小区的影响，因而大大地提高了控制的精度；4)由于其随机动态特征，使控制更有效和稳定。仿真显示我们的策略稳定地满足多业务QoS对呼叫中断概率的严格限制，同时又保证了信道的高吞吐量。

我们正在使用广义生灭过程模型对MBCA进行扩展，以适应未来的多业务呼叫对信道占有率的更大差异，从而进一步提高控制精度和无线资源的利用率。

参考文献

1 Rappaport T S. Wireless Communications: Principles and Practice. Englewood Cliffs, NJ: Prentice-Hall, 1996

2 Acampora A, Naghshineh M. Control and quality-of-service provisioning in high speed micro-cellular networks. IEEE Pers. Commun, 2nd quarter, 1994, 1: 36～43

3 Liu G, Zhu G, Wu W, Hu Z, Liu Y. An Effective Call Admission Policy for Multi-Services Wireless Networks Using Stochastic Control. In: Proc. IEEE Vehicular Technology Conf., 2004

4 Fang M, Chlamtac I, Lin Y.-B. Channel occupancy times and handoff rate for mobile computing and PCS networks. IEEE Trans. Comput., 1998, 47: 679～692

5 Oliveira C, Kim J B, Suda T. An adaptive bandwidth reservation scheme for high-speed multimedia wireless networks. IEEE J. Select. Areas Commun., Aug. 1998, 16: 858～874

6 Posner E, Guerin R. Traffic policies in cellular radio that minimize blocking of handoff calls. In: Proc. ITC-11, Kyoto, 1985. 294～298

7 Li B, Lin C, Chanson S. Analysis of a hybrid cutoff priority scheme for multiple classes of traffic in multimedia wireless networks. ACM/Baltzer J. Wireless Networks, Aug. 1998, 4: 279～290

8 Luo X, Li B, Thng I L-J, Lin Y-B, Chlamtac I. An Adaptive Measured-Based Preassignment Scheme With Connection-Level QoS Support for Mobile Networks. IEEE Trans. Wireless Commun. July 2002, 1: 512～529

9 Wu S, Wong K Y M, Li B. A new, distributed dynamic call admission policy for mobile wireless networks with QoS guarantee. IEEE/ACM Trans. Networking, Apr. 2002. 257～271

10 3GPP. 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects QoS Concept. 3GPP, 3G TR 23. 907 Version 1. 2. 0, 1999

11 Wang Zikun, Yang Xiangqun. Birth and Death Processes and Markov Chains. Berlin: Springer-Verlag, Beijing: Science Press, 1992. 220～237

12 Liu G, Zhu G, Wu W, Liu Y. An Adaptive Call Admission Policy for Broadband Wireless Multimedia Networks Using Stochastic Control. In Proc. IEEE WCNC, 2004, 3: 1324～1329

(上接第11页)

13 Tobies S. The complexity of reasoning with cardinality restrictions and nominals in expressive description logics. JAIR, 2000, 12: 199～217

14 Berners-Lee T. Weaving the Web. Harper, San Francisco, 1999

15 Berners-Lee T, Hendler J, Lassila O. The Semantic Web. Scientific American, 2001, 284(5): 34～43

16 Horrocks I, Patel-Schneider P F. Reducing OWL entailment to Description Logic satisfiability. ISWC-2003, Oct. 2003

17 Horrocks I, Patel-Schneider P F. Three Theses of Rep-resentation in the Semantic Web. WWW2003, Budapest, Hungary. ACM 1-58113-680- 3/03/0005, 2003

18 Horrocks I, Patel-Schneider P F, van Harmelen F. From *SHIQ* and RDF to OWL: The Making of a Web Ontology Language. Journal of Web Semantics, 2003

19 Patel-Schneider P F, Hayes P, Horrocks I. OWL Web Ontology Language Semantics and Abstract Syntax. W3C Recommendation. Feb. 2004. http://www. w3. org/TR/2004/REC-owl-semantics-20040210/

20 Available at: http://www. w3. org/2000/Talks/1206 xml2k-tbl/slide10-0. htmlwq

21 Horrocks I. Using an expressive description logic: FaCT or fiction? In: Proc. of the 6th Int. Conf. on Principles of Knowledge Representation and Reasoning(KR'98), 2001. 199～204

22 Haarslev V, Möller R. RACER system description. In: Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR'2001), volume 2083 of Lecture Notes in Artificial Intelligence, Spinger, 2001. 701～705

23 Savitch W J. Relationships between nondeterministic and deterministic tape complexities. Journal of Computer and System Sciences, 1970, 4(2): 177～192

24 Grosof B N, Horrocks I, Volz R, Decker S. Description logic programs: Combining logic programs with description logic. In: Proc. of the Twelfth International World Wide Web Conference (WWW 2003), ACM, 2003. 48～57