Steffen Staab, Rudi Studer. Handbook on Ontologies. Springer, 2004

Part I　Ontology Representation and Reasoning

1　Description Logics

描述逻辑

Franz Baader[1]，Ian Horrocks[2]，and Ulrike Sattler[1]

[1] Institut für Theoretische Informatik, TU Dresden, Germany

[2] Department of Computer Science, university of Manchester，UK

林伟明初译　曾新红审校

2006.4

[摘要] 在这一章，我们将阐述什么是描述逻辑以及为什么描述逻辑会产生良好的本体语言。特别地，我们将介绍描述逻辑 SHIQ，它构成了许多著名的本体语言(如 OIL，DAML+OIL 以及 OWL)的基础。我们认为，如果没有过去十年在描述逻辑上所做的基础研究，这一系列的知识表示语言就不可能在此领域扮演如此重要的角色。

描述逻辑推理在设计阶段和使用阶段均可使用。在设计阶段使用是为了提高本体的质量；在使用阶段使用是为了充分利用本体的丰富结构以及基于本体的信息。我们将展示如何用 tableaux 算法为描述逻辑如 SHIQ 提供可靠的，完备的推理，如何优化后实现使得这些服务可用到现实的应用中。我们还将讨论一些和 SHIQ 的扩展有关的挑战，这些扩展是由语言如 DAML+OIL 和 OWL 所要求的。最后，我们将简略地介绍一下新的推理服务如何能够支持描述逻辑知识库的建立。

1.1 引言

这一节的目的是简要地介绍描述逻辑以及讨论它们为什么非常适合作为本体语言。这一章余下的章节中，我们将通过提供更多的关于描述逻辑理论、推理算法以及实现技术这些技术细节来为这一骨架添加内容。关于这些以及其他与描述逻辑相关问题的更多的细节请看[9]。

本体

对于本体是什么有很多种定义，也许最著名的（至少在计算机科学家当中）是 Gruber 的定义：“本体是概念模型的明确的规格说明”[52][3]。在这里，概念模型是指客观世界的某一方面的抽象模型，表现为重要概念的性质以及它们之间关系的定义。明确的规范说明是指模型必须是用无二义性的语言来表示，使得机器如人一样能够顺利地处理它。

本体在知识管理、信息集成、协作信息系统、信息检索以及电子商务这些领域中变得越来越重要了。语义网[22]，是最近很热门的一个应用领域，在这里本体在代理之间建立一个通用的术语集从而确保不同的代理对于用于语义标记的术语都有着共用的理解。

本体的有效利用不仅仅需要一种设计良好的和定义明确的本体语言，还需要推理工具的支持。推理是很重要的，它既确保了本体的质量，又可以充分利用本体的丰富结构以及基于本体的信息。推理可以在本体生命周期的不同阶段使用。在本体的设计期间，推理可以用来测试概念是否没有矛盾，可以用来得出隐含的关系。特别地，人们通常需要计算概念级别，即，基于包含关系的指定概念的偏序关系。关于哪一个概念是另一个的特殊化以及哪些概念是同义词这些信息可用来在设计阶段测试该本体中的概念定义是否有预期的结果。这一信息在使用本体时也非常有用。

由于不可能假设所有的应用都使用同一个本体，不同本体之间的互操作性和集成也是个重要的问题。集成可以像这样来支持：在知识工程师断言了一些本体之间的关系之后，集成化的概念级别就会被计算并且检查概念的一致性。不一致的概念以及非预期的或者遗漏了的包含关系，就是错误的和不完全的本体间断言的信号，它们可以由知识工程师纠正和补充完全。

最后，在使用本体时也可以利用推理。如同使用预计算的概念级别一样，可以利用本体来确定在注解（annotation）中所声明事实的一致性，或者推断出注解实例和本体类之间的关系。更确切地说，当搜索注有来自本体的术语的 web 页时，不仅仅考虑精确的匹配，还考虑关于更通用的或者更专指的术语（后者的选择依赖于上下文）的匹配可能非常有用。但是，在使用阶段，对推理效率的要求比在设计和集成阶段更加严格。

在讨论为什么描述逻辑对于这样的本体语言是一个好的选择之前，我们将简单介绍一下描述逻辑。

描述逻辑

描述逻辑（DLs）[9,19,34]是知识表示语言的一种，它能以一种结构化的和形式上易懂的方式来表达一个应用领域的知识。一方面，领域中重要的观念是由概念*描述*（description）来进行表达的，即从原子概念（一阶谓词）和原子角色（二阶谓词）中利用由特定的描述逻辑所提供的概念和角色构造函数建立的表达式。另一方面，描述逻辑不同于它们的前辈如语义网络（semantic network）和框架，因为它们具备了形式化的、基于逻辑的语义。描述逻辑一词就是由此而来。

在这里，我们只通过一个例子来说明一些典型的构造函数。在 1.2 节将给出形式化的定义。假设我们要定义概念“一个与医生结婚的，至少有 5 个都是教授的孩子的男人”。这一概念可以用以下概念描述来进行表达：

$$\text{Human} \cap \neg\text{Female} \cap \exists\text{married.Doctor} \cap (\geq 5\ \text{hasChild}) \cap \forall\text{hasChild.Professor}$$

这一描述使用了布尔构造函数*合取*（$\cap$）（也可以理解为集合交），非（$\neg$）（也可以理解为集合补），以及*存在限制*构造函数（$\exists$R.C）,*值限制*构造函数（$\forall$R.C）,和*数字限制*构造函数（$\geq$ n R）。如果存在另外一个个体是与个体 Bob 结婚的（也就是说通过 married 角色与 Bob 相关），并且该个体是一个医生（也就是说，属于概念 Doctor），那么个体 Bob 就属于$\exists$married.Doctor。同样地，Bob 属于($\geq$ 5 hasChild) 当且仅当他有至少 5 个孩子，Bob 属于$\forall$hasChild.Professor 当且仅当他所有的孩子（即，所有通过 hasChild 角色与 Bob 相关的个体）都是教授。

除了这种描述形式体系之外，描述逻辑通常配备有一个术语形式体系和一个断言形式体系。以其最简单的格式，*术语公理*（terminological axiom）可以用来为复杂描述引进名称（缩写）。例如，对于上面介绍的概念描述，我们可以引进缩写 HappyMan。更富有表达力的术语形式体系允许限制语句，如

$$\exists\text{hasChild.Human} \subseteq \text{Human},$$

意思是只有人类才能有是人类的孩子。*断言形式体系*（assertional formalism）可以用来声明个体的属性。如，断言

$$\text{HappyMan(BOB)}, \text{hasChild(BOB,MARY)}$$

声明 Bob 属于概念 HappyMan，Mary 是 Bob 的其中一个孩子。

描述逻辑系统为他们的用户提供了多种推理能力，从显式表示的知识中推导出隐含的知识。*包含*（subsumption）算法确定子概念和超概念关系：C 被包含于 D，当且仅当 C 中所有的实例都必然是 D 中的实例，也就是说，第一个描述总是被解释为第二个描述的子集。如，上面给出的概念 HappyMan 的定义，HappyMan 被包含于$\exists$hasChild.Professor，因为

HappyMan 的实例至少有 5 个孩子，他们都是教授，也就是他们也有一个孩子是教授。*实例*（instance）算法确定了实例关系：个体 i 是概念描述 C 的一个实例，当且仅当 i 总是被解释为 C 的一个元素。例如，上面给出的断言和 HappyMan 的定义，MARY 是 Professor 的一个实例。*一致性*（consistancy）算法确定一个知识库（由一组断言和一组术语公理组成）是否没有矛盾。例如，如果我们把¬Professor(MARY)加到上述的两个断言中，那么包含这些断言和 HappyMan 的定义的知识库是不一致的。

为了确保描述逻辑系统的合理的和可以预测的性能，对于系统所使用的描述逻辑，这些推理问题必须至少是可判定的，并且最好是低复杂度的。因此，有问题的描述逻辑的表达能力必须以合适的方式来进行限制。但是，如果强加的限制过于严格，则不能够表达出应用领域的重要概念。在描述逻辑研究中，研究描述逻辑的表达能力和它们的推理问题的复杂度之间的权衡已经成为最重要的问题之一。与这个问题有关的研究可以粗略地分成以下四个阶段：

阶段一（1980-1990）主要与系统的实现有关，如 KLONE,K-REP,BACK 和 LOOM[28，77，87，76]。这些系统使用了所谓的*结构化包含算法*（structural subsumption algorithm），该算法第一次规范化了概念描述，并且递归比较了规范化描述的语法结构[79]。这些算法通常相对有效（多项式的），但他们有个缺点，只对于非常无表达能力的描述逻辑是完备的，也就是说，对于更有表达能力的描述逻辑，他们不能检测出所有存在的包含/实例关系。这一阶段的后期，对于描述逻辑中推理复杂度的早期正式研究表明，大多数描述逻辑并没有多项式时间推理问题[27，80]。为此，CLASSIC 系统（第一个工业强度的 DL 系统）的实现小心地限制了他们的描述逻辑的表达能力[86，26]。

阶段二（1990-1995）开始于将一个新的算法范例引入描述逻辑中，该算法称为*基于 tableau 的算法*（tableau-based algorithm）[94，42，60]。他们对于命题封闭描述逻辑（即，带有全部布尔运算符的描述逻辑）有效，并且对于表达能力强的描述逻辑是完备的。为了判定一个知识库的一致性，基于 tableau 的算法试图通过分解知识库中的概念来建立该知识库的一个模型，从而推出在该模型元素上的新的约束。该算法的停止有两个原因可引发，一个是因为对于建立模型的所有尝试均因出现明显的矛盾而失败，另一个是因为“正则”模型。因为在命题封闭的描述逻辑中，包含和可满足性可归结为一致性的判别，一致性判别算法能解决上述提到的所有推理问题。首批使用该算法的系统（KRIS 和 CRACK）证明，即使相关推理问题的最坏情况时间复杂度不再是多项式时间[13，30]，这些算法的优化实现也会导致系统的一个可接受的性能。这一阶段还出现了多种描述逻辑中的推理复杂度的一个彻底分析[42，43，41]。另外一个重要发现是描述逻辑与模态逻辑非常相关[92]。

阶段三（1995-2000）表现为对于非常有表达能力的描述逻辑，不管是基于 tableau 方法的[66，67]，还是基于转换为模态逻辑的[38，39，37，40]，研制出了许多推理程序。高度优化系统（FaCT，RACE 和 DLP[62，53，85]）表明，用于有表达能力的描述逻辑的基于 tableau 的算法可以导致系统的一个很好的实用性能，甚至在（一些）大型的知识库上也是如此。在这一阶段，也更详细地研究了与模态逻辑和与先序逻辑的可判定段之间的关系 [23，83，50，48，49]，以及研究了在数据库中的应用（如模式推理，查询优化以及数据库集成）[31，33，35]。

我们现在处于*第四阶段*的开始，在这一阶段，开发出使用非常有表达能力的描述逻辑以及基于 tableau 算法的工业强度描述逻辑系统，并且还有一些应用，如语义网，或大脑生物信息的知识表示和集成。

描述逻辑作为本体语言

前面已经提到，对于许多应用，高质量的本体是至关重要的，他们的构造，集成以及演化很大程度依赖着定义良好的语义和强大的推理工具。由于描述逻辑提供了上述两个方面，对于本体语言，它应该是个理想的选择。这在 10 年前已经相当清晰了，但是在那时，描述逻辑系统所提供的表达能力和推理有效性，与本体工程师所需要的表达能力和大型知识库之间基本上是不匹配的[44]。通过我们前面概括的过去 10 年对描述逻辑所做的基础研究，最后本体工程师的需要与描述逻辑研究者所提供的系统之间的间隙变得窄得可以在上面建一座稳定的桥梁。

作为许多 web 本体语言（包括 OIL[46]，DAML+OIL[63，65]和 OWL——由 W3C Web-本体工作组（http://www.w3.org/2001/sw/WebOnt/）所开发的一种新的本体语言）的基础，描述逻辑作为本体语言的适用性已显得非常突出。所有这些语言的语法都基于 RDF Schema，但它们的设计基础是有表达能力的描述逻辑 SHIQ[68][5]，并且，开发者试图在表达能力以及推理复杂度之间寻找一个好的折衷。尽管 SHIQ 中的推理是可判定的，但它还是有相当高的最坏情况复杂度（指数时间）。尽管如此，高度优化的 SHIQ 推理机（如 FaCT[62]和 RACER[55]）在实际应用中表现得非常优秀。

让我们指出 SHIQ 的一些特性，它们使得描述逻辑有足够的表达能力以至可以用来作为一种本体语言。首先，SHIQ 提供数字限制，它比之前提到的那些 DL（在早期描述逻辑系统中使用）更加有表达能力。利用 SHIQ 提供的*受限数字限制*（qualified number restriction），人们可以表达一个人有最多两个孩子（并没有提及这些孩子的属性）：

$$(\leq 2\ \text{hasChild})$$

也可以指定至多有一个儿子和至多有一个女儿：

$$(\leq 1\ \text{hasChild}.\neg\text{Female}) \cap (\leq 1\ \text{hasChild}.\text{Female})$$

第二，SHIQ 允许复杂的术语公理的公式化，如“人类有是人类的双亲”：

$$\text{Human} \subseteq \exists \text{hasParent}.\text{Human}.$$

第三，SHIQ 同样考虑了*逆角色*（inverse role），*传递角色*（transitive role）以及*子角色*（subrole）。例如，除了用 hasChild 之外也可以用它的逆 hasParent, 还可规定 hasAncestor 是传递的，并且 hasParent 是 hasAncestor 的子角色。

描述逻辑和本体界已经证明，当描述聚集对象的性质和建立本体的时候这些特性起到了关键的作用[90，96，45]。提供这些特性的 DL 作为 web 本体语言 OIL，DAML+OIL 和 OWL 的底层逻辑形式方法被实际应用证明了这一观点[96][6]。

最后，我们简单地提一下通常用在本体语言中的 SHIQ 的两个扩展（我们将在 1.7 节进行详细讨论）。

具体域（Concrete domain）[11，74]使得描述逻辑与具体集合（如实数，整数或字符串）、内在的谓词（如比较关系≤，带常数的比较关系≤17 或者 isPrefixOf）相结合。这支持抽象对象的具体性质的建模，如一个人的年龄、体重或名字，以及这些具体性质的比较。但是，由于它们这种无限制的形式，具体域可能在底层描述逻辑的可判定性和计算复杂度上产生显著的影响[74]。

名词性词（Nominal）是特定概念名称，它们被解释为单元素集合。使用名词 Turing，我们可以用 CSientist ∩ ∃hasMet.Turing 描述所有遇见 Turing 的计算机科学家。同样，名词性词也可能对一个逻辑的复杂度产生显著的影响。

1.2 富有表达能力的描述逻辑 SHIQ

在本节，我们给出富有表达力的 DL SHIQ 的语法和语义（尽管 OWL 下的 DL 在某些方

面更富有表达能力一点——请看 1.4 节）。此外，我们会着重介绍*术语*（terminological）形式体系，即，支持对应用领域中的相关概念进行定义和对限制这些概念的解释的约束进行声明的部分。由于篇幅的原因，并且*断言*（assertional）形式体系在本体工程中只起到次要的作用，在此就不介绍了。有兴趣的读者可以参考[9，91]来大体了解断言的描述逻辑形式体系，并可参考[54，69]来了解 SHIQ 的断言推理。

对比在文献中提到的、只关注用来定义概念的构造函数的大部分描述逻辑，SHIQ[67]还考虑到有相当表达能力的角色。当然，这些角色可用来定义概念。我们先来定义 SHIQ 角色，然后再来定义 SHIQ 概念。

定义 1（SHIQ 角色的语法和语义）：令 R 为角色名称的集合，它划分为传递角色的集合 R_+ 和标准角色集合 R_P。所有的 SHIQ 角色的集合是 $R \cup \{r^- \mid r \in R\}$，这里 r^- 是指角色 r 的逆。一个角色包含公理是 $r \subseteq s$ 形式的，这里的 r , s 都是 SHIQ 角色。一个角色层级是角色包含公理的有限集合。

解释 $I=(\triangle^I, \cdot^I)$ 由集合 $\triangle^I$ 与函数 $\cdot^I$ 组成，$\triangle^I$ 称为 I 的域，$\cdot^I$ 则把每一个角色影射到 $\triangle^I \times \triangle^I$ 的子集使得对于所有的 $p \in R$ 和 $r \in R_+$,

$$\langle x, y\rangle \in p^I \text{ 当且仅当 } \langle y, x\rangle \in (p^-)^I$$

$$\text{如果} \langle x, y\rangle \in r^I \text{ 且 } \langle y, z\rangle \in r^I \text{ 那么} \langle x, z\rangle \in r^I$$

解释 I 满足角色层级 R 当且仅当对于每一个 $r \subseteq s \in R$，$r^I \subseteq s^I$；这样的解释称为 R 的一个模型。

这些角色在 SHIQ 的所有概念构造函数（后面将会定义）中的无限制使用会导致一个不可判定的描述逻辑[67]。因此，我们必须定义一个所有 SHIQ 角色的合适子集。这需要更多的概念。

1. 二元关系上的逆关系是对称的，也就是说，r^- 的逆还是 r。为了避免如 r^{--} r^{---} 这样来书写角色表达式，我们定义一个函数 Inv, 该函数返回的是角色的逆：

$$\text{Inv}(r) := \begin{cases} r^- & \text{如果 r 是角色名} \\ s & \text{如果对于角色名 s，} r = s^- \end{cases}$$

2. 由于集合包含关系是传递的，并且两个角色的包含关系转到它们的逆中，一个给定的角色层级 R 隐含着额外的包含关系。为了说明这一事实，我们定义 $\subseteq_R^*$ 作为自反-传递闭包：

$$\subseteq_R^* := R \cup \{Inv(r) \subseteq Inv(s) \mid r \subseteq s \in R\}$$

我们用 $r \equiv_R s$ 作为 $r \subseteq_R^* s$ 和 $s \subseteq_R^* r$ 的缩写。假若这样，R 的每一个模型会把这些角色解释为同样的二元关系。

3. 显然，一个二元关系是传递的当且仅当它的逆是传递的。因而，如果 $r \equiv_R s$ 并且 r 或者 Inv(r)是传递的，那么 R 中的任意一个模型都会把 s 解释为一个传递二元关系。为了说明这种隐含的传递角色，我们定义函数 Trans:

$$\text{Trans}(s, R) := \begin{cases} \text{true} & \text{对于一些有关系 } r \equiv_R s \text{ 的 r，如果 } r \in R_+ \text{ 或者 } \text{Inv}(r) \in R_+ \end{cases}$$

false 其他情况

4．角色 r 称为是关于 R 简单的当且仅当对于所有 $s \sqsubseteq_R r$ 都有 Trans(s, R) = false。

定义 2（SHIQ 概念的语法和语义）。令 Nc 为概念名称的集合。SHIQ 概念的集合是最小的集合使得：

1．任一概念名称 $A \in Nc$ 是一个 SHIQ 概念

2．如果 C 和 D 是 SHIQ 概念并且 r 是一个 SHIQ 角色，那么 $C \cap D, C \cup D, \neg C, \forall r.C$ 和 $\exists r.C$ 都是 SHIQ 概念。

3．如果 C 是 SHIQ 概念，r 是一个简单的 SHIQ 角色，并且 $n \in N$, 那么 $(\leq n\, r.C)$ 和 $(\geq n\, r.C)$ 都是 SHIQ 概念。

解释 $I=(\triangle^I, \cdot^I)$ 的解释函数 $\cdot^I$ 又把每一个概念影射到 $\triangle^I$ 的子集中使得：

$(C \cap D)^I = C^I \cap D^I$, $(C \cup D)^I = C^I \cup D^I$, $\neg C^I = \triangle^I \setminus C^I$,

$(\exists r.C)^I = \{ x \in \triangle^I \mid$ 存在某个 $y \in \triangle^I$ 有 $\langle x, y\rangle \in r^I$ 且 $y \in C^I \}$,

$(\forall r.C)^I = \{ x \in \triangle^I \mid$ 对于所有 $y \in \triangle^I$，如果 $\langle x, y\rangle \in r^I$，那么 $y \in C^I \}$,

$(\leq n\, r.C)^I = \{ x \in \triangle^I \mid \#\, r^I(x, C) \leq n \}$,

$(\geq n\, r.C)^I = \{ x \in \triangle^I \mid \#\, r^I(x, C) \geq n \}$,

这里的#M 是集合 M 的基数,并且有 $r^I(x, C) := \{y \mid \langle x, y\rangle \in r^I$ 且 $y \in C^I\}$。如果 $x \in C^I$,那么我们说 x 是 I 中的概念 C 的一个实例；如果 $\langle x, y\rangle \in r^I$,那么 y 被称为 I 中 x 的一个 r 后继。

概念可以用来描述一个应用领域的相关概念。术语(TBox)为复杂的概念引进了缩写词（名称）。 在 SHIQ 中，TBox 同样允许声明更复杂的约束。

定义 3. 一般概念包含（GCI）的形式是 $C \subseteq D$，其中 C, D 都是 SHIQ 概念。GCI 的一个有限集合称为 TBox。解释 I 是 TBox T 的一个模型当且仅当它满足在 T 中的所有 GCI，也就是说，对于任意的 $C \subseteq D \in T$ 均有 $C^I \subseteq D^I$。

一个概念定义的形式是 $A \equiv C$，其中 A 是一个概念名称。它可看成是两个 GCI $A \subseteq C$ 和 $C \subseteq A$ 的缩写。

推理问题是关于一个 TBox 和一个角色层级来进行定义的。

定义 4. 概念 C 关于角色层级 R 和 TBox T 是可满足的，当且仅当存在一个 R 和 T 的模型 I 并且有 $C^I \neq \phi$。这样的解释叫做 C 关于 R 和 T 的一个模型。概念 D 关于<R,T>包含概念 C(写为 $C \subseteq_{<R,T>} D$)当且仅当对于 R 和 T 的所有模型 I 都有 $C^I \subseteq D^I$。概念 C,D 是关于<R,T>等价的（写成 $C \equiv_{<R,T>} D$）当且仅当他们互相包含。

根据定义，等价可归结为包含。此外，包含可归结为可满足性因为 $C \subseteq_{<R,T>} D$ 当且仅当 $C \cap \neg D$ 是关于 R 和 T 不可满足的。

正如前面所说，大部分描述逻辑是（一阶）谓词逻辑的（可判定）段[23，1]。把角色名称看成是二元关系，把概念名称看成是一元关系，例如，角色包含公理 $r \subseteq s^-$ 可转化为

$\forall x \forall y.\ r(x, y) \Rightarrow s(y, x)$, 并且 GCI $A \cap \exists r.\ C \subseteq D \cup \forall\, s^-.E$ 可转化为

$\forall x.(A(x) \wedge \exists\, y.\ r(x, y) \wedge C(y)) \Rightarrow (D(x) \vee \forall y.\ s(y, x) \Rightarrow E(x))$

这样的转换保存了语义：我们可以很容易地把描述逻辑的解释看成是谓词逻辑的解释，并且证明，举个例来说，关于 TBox T 和角色层级 R 的概念 C 的任何一个模型都是 C 的转换与 T 和 R 的（全称量化）转换相结合的模型。

在介绍如何解决 SHIQ 中的可满足性问题之前，我们先来看一看如何用 SHIQ 来定义本体。

1.3 用 SHIQ 来描述本体

一般来说，一个本体可像下面这样形式化为一个 TBox。首先，我们通过在允许的解释上引进约束来限制可能的世界。例如，要表达“在我们的世界中，我们要考虑人，他们或者是麻瓜（muggle，不懂魔法的普通人）或者是巫师（sorcerer）”，我们可以用 GCI 来表达

$$\text{Human} \subseteq \text{Muggle} \cup \text{Sorcerer} \quad \text{并且} \quad \text{Muggle} \subseteq \neg\, \text{Sorcerer}$$

接下来，我们可以用如下的 GCI 来表达“人正好有一对双亲，并且人的所有双亲和孩子都是人”：

$$\text{Human} \subseteq \forall \text{hasParent}.\, \text{Human} \cap (\leq 2\ \text{hasParent}.\top) \cap (\geq 2\ \text{hasParent}.\top) \cap \forall \text{hasParent}^{-}.\text{Human}$$

这里的 T 是顶级概念 $A \cup \neg A$ 的缩写。（当限制概念是 T 时，就等同于一个无条件的限制，常常写成 $\leq 2\ \text{hasParent}$）

此外，我们考虑传递角色 hasAncestor, 和角色包含

$$\text{hasParent} \subseteq \text{hasAncestor}$$

接下来的 GCI 表达的是有祖先是巫师的人本身也是巫师：

$$\text{Human} \cap \exists \text{hasAncestor}.\text{Sorcerer} \subseteq \text{Sorcerer}.$$

第二，我们可以利用概念定义来定义我们应用领域的相关思想。回想一下概念定义 $A \equiv C$ 代表着两个 GCI: $A \subseteq C$ 以及 $C \subseteq A$。如果一个概念名称在概念定义的左边，那么它被称为*定义的*（defined），否则称为*原语*（primitive）。

我们希望我们的概念定义有明确的效果，也就是说，原语概念和角色名称的解释必须唯一决定定义的概念名称的解释。为此，带有额外 GCI 概念定义的集合必须满足以下三个条件：

1. 没有多重定义，即，每个定义的概念名称必须最多只能在概念定义的左边出现一次。
2. 没有循环定义，即，在概念定义的集合中，定义的名称之间没有循环依赖。（为了给循环定义明确的效果，人们需要给他们使用定点语义[81，4]）
3. 定义的名称不出现在任何额外的 GCI 中。

与概念定义相反，SHIQ 中的 GCI 可能在概念名称之间存在循环依赖。如上面描述人 human 的例子。

作为满足上述限制的概念定义集合的一个简单例子，我们来定义概念 grandparent 和 parent（除了将孩子与他们的父母联系起来的角色 hasParent，我们还使用概念 Parent 来描述所有有孩子的人类）：

$$\text{Parent} \equiv \text{Human} \cap \exists \text{hasParent}^{-}.\top$$

$$\text{Grandparent} \equiv \exists \text{hasParent}^{-}.\text{Parent}$$

由以上的概念定义和 GCI 组成的 TBox 以及 hasAncestor 是 hasParent 的传递超角色的事实，意味着如下的包含关系：

$$\text{Grandparent} \cap \text{Sorcerer} \subseteq \exists \text{hasParent}^{-}.\exists \text{hasParent}^{-}.\text{Sorcerer},$$

即，是巫师的祖父有个是巫师的孙子。尽管这样的结论给定假设后可能听起来很有道理，但它还是需要相当多的推理去获取。特别地，必须用到以下事实：hasAncestor 是传递的(has Ancestor^{-}也是传递的)，hasParent^{-}是 hasParent 的逆，并且我们有 GCI 说人类的孩子还是人类。

总的来说，一方面 SHIQ-TBox 可以通过 GCI、传递声明、角色包含来使一个应用领域（原语概念）的基本概念公理化，从某种意义上来说这些声明约束了基本概念（notion）的可能解释。另一方面，可通过概念定义引进更加复杂的概念（notion）（定义的概念）。给定基本概念的一个解释，概念定义唯一地确定了定义的概念的解释。

这样一个 TBox 的*分类表*（taxonomy）由定义概念的包含层级所给出。它可通过一个用于 SHIQ 的包含算法来计算（请看接下来的 1.5 节）。知识工程师可以通过检测定义概念的可满足性（因为为空概念给出复杂的定义是没有意义的）和检测它们在分类表中的位置是否对应着直觉中的位置，来测试该 TBox 是否捕获到了她的直觉知识。SHIQ 丰富的表达能力以及人们可以如上面所说的那样“验证”TBox 的事实，是 SHIQ 为何非常合适作为本体语言的主要原因[90，45，96]。

1.4 SHIQ 和 OWL

正如已经讨论过的，OWL 是一种语义网本体语言（基于 DAML+OIL，由 W3C Web-本体工作组开发），它的语义可以通过转换成富有表达力的描述逻辑来定义。这并不是一种巧合——而是一个设计目标。这种影射允许 OWL 充分利用来自描述逻辑研究的形式化成果（例如关于关键推理问题的可判定性和复杂度）以及利用已经实现了的描述逻辑推理机（如 FaCT[61]和 RACER[55]）来为 OWL 应用提供推理服务。

一个 OWL（Lite 或 DL）本体可以看成是对应着带有角色层级的描述逻辑 TBox，利用类（class）（对应于概念）和属性（property）（对应于角色）来描绘领域。一个本体由一组公理组成，这些公理断言了，如，类之间或者属性之间的包含关系。

构造函数	描述逻辑语法	例子
intersectionOf	$C_1 \cap \ldots \cap C_n$	Human $\cap$ Male
unionOf	$C_1 \cup \ldots \cup C_n$	Doctor $\cup$ Lawyer
complementOf	$\neg C$	$\neg$Male
one of	$\{x_1 \ldots x_n\}$	{john, mary}
allValuesFrom	$\forall P.C$	$\forall$hasChild.Doctor
someValuesFrom	$\exists r.C$	$\exists$hasChild.Lawyer
hasValue	$\exists r.\{x\}$	$\exists$cityzenOf.{USA}
minCardinality	$(\geq n\ r)$	($\geq$ 2 hasChild)
maxCardinality	$(\leq n\ r)$	($\leq$ 1 hasChild)
inverseOf	r^{-}	hasChild $^{-}$

图 1.1 OWL 构造函数

正如在一个标准的描述逻辑中，OWL 类可以是名称或者利用各种各样的构造函数从简单的类或属性建立起来的表达式。图 1.1 概括了 OWL 所支持的构造函数及其等价的描述逻辑抽象语法。RDF 语法的完全 XML 序列化并没有在此展示出来，因为它太冗长了，如，

Human ∩ Male 可写成：

```
<owl:Class>
    <owl:intersectionOf  rdf:parseType=”Collection”>
        <owl:Class  rdf:about=”#Human” />
        <owl:Class  rdf:about=”#Male” />
    </owl:intersectionOf>
</owl:Class>
```

而(≥ 2 hasChild.Thing)可写成：

```
<owl:Restriction>
    <owl:onProperty  rdf:resource=”#hasChild” />
        <owl:minCardinality
            rdf:datatype=”&xsd;NonNegativeInteger” >2
        </owl:minCardinality>
    </owl:Restriction>
```

像 owl:和&xsd;这样的前缀为资源指定了 XML 命名空间，而 rdf : parseType=”Collection” 则是 RDF 的一个扩展，为使用带有 first 属性和 rest 属性（rest 属性可以消除，但会变得很冗长）的三元组定义的 lisp 样式表提供简写符号。例如，上面的第一个例子由如下的三元组组成：<r_1, owl:intersectionOf, r_2>, <r_2, owl:first , Human>, <r_2, rdfs:type, Class>, <r_2, owl:rest, r_3>等等，这里，r_i 是一匿名资源，Human 代表命名资源“Human”的统一资源标识符 URI，owl:intersectionOf , owl:first , owl:rest 和 rdfs:type 代表命名所述属性的统一资源标识符。

OWL 的一个重要特征是，除了可使用由本体所定义的“抽象”类之外 ，还可以在 someValuesFrom, allValuesFrom, 和 hasValue 约束中使用 XML Schema *数据类型*（如 string, decimal 和 float）。例如，类 Adult 可断言成等同于 Person ∩ ∃age.over17, 这里的 over17 是一种基于 decimal 的、带有附加约束表明其值必须至少是 18 的 XML Schema 数据类型。利用 XML Schema 与 RDF 的结合可把它写成如下形式：

```
<xsd:simpleType name = “over17”>
    <xsd:restriction base = “xsd:positiveInteger”>
    <xsd:minInclusive value = “18”/>
    </xsd:restriction>
</xsd:simpleType>

<owl:Class rdf:ID = “Adult”>
    <owl:intersectionOf rdf:parseType = “Collection”>
        <owl:Class rdf:about = “#Person”/>
        <owl:Restriction>
            <owl:onProperty rdf:resource = “#age” />
            <owl:someValuesFrom rdf:resource = “#over17” />
        </owl:Restriction>
    </owl:intersectionOf>
</owl:Class>
```

正如前面已提到的，一个 OWL 本体由一组公理组成。图 1.2 概括了 OWL 所支持的公理。这些公理使得它可以断言关于类或属性的包含或等价，类的不相交性以及个体（资源）的等价或不等价。此外，OWL 也允许断言属性（即，描述逻辑角色）的属性。特别地，可

以断言一个属性是传递的，函数的，逆函数的或者对称的。

Axiom	DL Syntax	Example
subClassof	$C_1 \subseteq C_2$	Human $\subseteq$ Animal $\cap$ Biped
equivalentClass	$C_1 \equiv C_2$	Man $\equiv$ Human $\cap$ Male
subPropertyOf	$P_1 \subseteq P_2$	hasDaughter $\subseteq$ hasChild
equivalentProperty	$P_1 \equiv P_2$	cost $\equiv$ price
disjointWith	$C_1 \subseteq \neg C_2$	Male $\subseteq \neg$Female
sameAs	$\{x_1\} \equiv \{x_2\}$	{President_Bush} $\equiv$ {G_W_Bush}
differentFrom	$\{x_1\} \subseteq \neg\{x_2\}$	{john} $\subseteq \neg$ {peter}
TransitiveProperty	$P \in R_+$	$\text{hasAncestor}^+ \in R_+$
FunctionalProperty	$T \subseteq (\leq 1\ P)$	$T \subseteq (\leq 1$ hasMother)
InverseFunctionalProperty	$T \subseteq (\leq 1\ P^-)$	$T \subseteq (\leq 1$ isMotherOf $^-)$
SymmetricProperty	$P \equiv P^-$	isSiblingOf $\equiv$ isSiblingOf$^-$

图 1.2. OWL 公理

这表明，除了个体和数据类型之外，OWL 的构造函数和公理均可以转换成 SHIQ。事实上，OWL 等价于带有名词性词和一个简单形式的具体领域的 SHIN（只有 T 概念允许受限数字限制的 SHIQ）的扩展。（将在 1.7 节详细地讨论这一扩展）。

1.5 SHIQ 中的推理

在 1.2 节我们专注于*术语的*（terminological）的推理，也就是，关于 TBox（即通用概念包含集合）和角色层级的可满足性和包含。像关于一个 TBox 的 ABox 一致性判断这样的断言推理问题可以用非常类似的技术来进行判断[91，9，54，69]。正如 1.2 节所说，包含可以（在线性时间）归约为可满足性。此外，因为 SHIQ 允许子角色和传递角色，TBox 可以被内在化，也就是说，关于 TBox 和角色层级的可满足性可归约为关于空的 TBox 和角色层级的可满足性。原则上，这可以通过引进一个（新的）传递的超角色 u（它是出现在 TBox 中的所有角色的超角色）以及概念 C_0（用来测试可满足性）来完成。那么把 C_0 扩展为概念

$$\hat{C}_0 := C_0 \cap \bigcap_{C \subseteq D \in T} (\neg C \cup D) \cap \forall u.(\neg C \cup D)。$$

然后我们可以证明 $\hat{C}_0$ 关于扩展了的角色层级是可满足的当且仅当原概念 C_0 关于 TBox T 和原角色层级是可满足的[2，92，8，67]。

因此，可以设计一个算法来判定 SHIQ 概念关于角色层级和传递角色的可满足性。这一问题被称为 EXPTIME-complete[97]。事实上，可以通过对在命题动态逻辑上的可满足性的 EXPTIME-hardness 证明进行简单改写来证明 EXPTIME-hardness [47]。利用基于自动机的方法，Tobies[97]证明了关于角色层级的 SHIQ 概念的可满足性确实是在指数时间内可判定的。

在本节余下的内容中，对于此问题我们介绍一个基于 tableau 的判定程序。这个程序（在[67]中有详细介绍）以最坏情况*非确定*（nondeterministic）双指数时间运行 [13]。然而，根据目前的技术水平，这个程序比在[97]中的基于自动机的 EXPTIME 程序要更加实用。事实上，

这是描述逻辑系统 FaCT[62]的高度优化实现的基础。

从一个 SHIQ 概念 C_0、一个角色层级 R 和“角色是传递的”信息着手，该算法试图构造一个关于 R 的 C_0 的模型。由于 SHIQ 有所谓的树模型属性，我们可以假定这个模型具有一个无限树的形式。如果我们要得到一个判定程序，我们只能构造一棵有限树来代表那棵无限树（如果该树模型存在）。这可以做得到，这样这个有限表示就可以*拆成*（unravel into）一个关于 R 的 C_0 的无限树模型 I。在表示该模型的有限树中，一个节点 x 对应一个个体 $\pi(x) \in \Delta^I$，并且我们用一组概念 L(x) 来标示每个节点，假设 $\pi(x)$ 是 L(x)的一个实例。同样，边表示角色-后继关系，在 x 与 y 之间的边用（假设）连接 x 与 y 的角色来标示。该算法要么终止于一个树模型的有限表示的形成，要么因*冲突*而终止，也就是说，遇到一个明显的不一致，如$\{C, \neg C\} \subseteq L(x)$。前者表示 C_0 关于 R 是可满足的，后者表示 C_0 关于 R 是不可满足的。

该算法以一棵含有以 $L(x) = \{C_0\}$标示的简单节点 x 的树来初始化。然后它应用所谓的*完备规则*（completion rule）依照句法分解节点标示上的概念，从而为给定的节点推理出新的约束，然后根据这些约束来扩展树。例如，如果 $C_1 \cap C_2 \in L(x)$，那么∩规则把 C_1 和 C_2 都添加到 L(x)中。如果$(\geq n\ r.C) \in L(x)$和 x 都还没有 n 个标示中带有 C 的不同的 r 后继，那么≥规则产生 x 的 n 个新的 r 后继节点 $y_1,\ldots,y_n$,并标示上 $L(y_i) = \{C\}$。此外，它断言这些新的后继必须保持独立（即，在该算法的后续步骤中不能被等同起来）。其他规则更加复杂，对该算法的完整描述超出了本章的范围。但是，我们想指出两个问题，它们使 SHIQ 中的推理比在更弱表达能力的描述逻辑中的推理困难得多。

首先，与在大多数早期描述逻辑系统中使用的非受限的数字限制相比，*受限的数字限制*更加难以处理[14]（注意，和 DAML+OIL 不同，OWL 只支持非受限的数字约束）。让我们用例子来说明它。假设算法产生了一个节点 x，它带有标示$(\leq 1\ \text{hasChild.T}) \in L(x)$,并且它有两个 hasChild 后继 y_1,y_2（即，两条标示了 hasChild 的分别指向 y_1,y_2 的边)。为了满足对于 x 的数字限制$(\leq 1\ \text{hasChild.T})$，算法把节点 y_1 等同于节点 y_2(除非这些节点已被断言为不同的节点，在这种情况下我们会有一个冲突)。现在假设我们还有个节点 x，它带有两个 hasChild 后继 y_1,y_2，但是 x 的标示包含一受限的数字限制如$(\leq 2\ \text{hasChild.Parent})$。一个幼稚的想法[98]是检测 y_1 和 y_2 的标示是否含有 Parent,并且只有两者都含有这个概念时才把 y_1 和 y_2 等同起来。然而，这并不正确，因为从树中构造出的模型 I, $\pi(y_i)$也很可能属于 Parent^I，即使该概念并不属于 x 的标示。第一个能处理受限数字限制的正确的算法在[59]中被提出。主要思想是引进一个所谓的 choose 规则。在我们的例子里，这个规则将（非确定地）选择 y_i 是属于 Parent 还是¬ Parent，并相应地扩展它的标示。如果加上选择规则，上述的幼稚的等同规则实际上是正确的。

其次，在传递角色面前，保证算法终止并非是微不足道的任务[56，88]。如果对于传递角色 r 有$\forall r\ .\ C \in L(x)$,那么我们不仅要把 C 添加到 x 的任何 r 后继 y 的标示上，还要把$\forall r\ .\ C$添加上去。这确保，即使在“r 链”上

$$x \xrightarrow{r} y \xrightarrow{r} y_1 \xrightarrow{r} y_2 \xrightarrow{r} \ldots \xrightarrow{r} y_n$$

我们的确得出 $C \in L(y_n)$。这是必要的，因为从由算法产生的树而构造出来的模型中，有

$$(\pi(x),\pi(y)), (\pi(y),\pi(y_1)), \ldots, (\pi(y_{n-1}),\pi(y_n)) \in r^I,$$

这样一来 r^I 的传递性也要求$(\pi(x),\pi(y_n)) \in r^I$。 因而，在 x 上的值限制也同样应用到 y_n 上. 通过 r 边传输$\forall r\ .\ C$ 以确保它被照顾到。然而，这也可能会导致非终止。例如，考虑概念$\exists r.A \cap \forall r.\exists r.A$ 其中 r 是传递角色。易知算法会产生一个无限链的带有$\{A, \forall r.\exists r.A, \exists r.A\}$标示的节点。为了防止这循环并确保终止，我们利用一个叫做 blocking 的循环检测机制：如果节点 x

的标示与它的祖先之一相同，我们“中断”对 x 进行规则应用。中断的条件必须公式化，这样不管中断什么时候发生，我们都可以“解开”中断（有限）路径成为构造出来的模型中的一条无限路径。在描述逻辑中，中断首次在[3]中的一个算法的上下文中使用，该算法可以处理角色的传递闭包，并且在[7，32，6]中得到改进。在 SHIQ 中，中断条件是相当复杂的，因为带有数字限制的传递与逆角色的结合需要一个相当先进的解开形式[67]。事实上，这种构造器的结合对以下事实是重要的：因为和在文献中考虑的大多数描述逻辑不同，SHIQ 并没有有限的模型属性，也就是说，存在可满足的 SHIQ 概念，它们只在无限解释中是可满足的[67]。

1.6 SHIQ 的实际推理服务

正如 1.5 节提到的，SHIQ 中的推理是高复杂度的（指数）。导致如此高的*最坏情况*复杂度的病态情况有点人为性，极少出现在实践中[80，58，95，62]。但是，即使在实际应用中，也会出现那些很难用理论算法的幼稚实现来解决的问题，如在 1.5 节所概述的那个算法。因此，现代描述逻辑系统包含大范围的优化技术，已经证明利用它们能改进典型情况性能好几个数量级[64]。这些系统展示了很好的典型情况性能，并且在实际应用中运行良好[10，30，62，55，85]。

优化技术的详细介绍超出了本章范围，有兴趣的读者可以参考[9]以获得更多的信息。但概述一些关键技术如惰性展开（lazy unfolding），吸收（absorption）以及相关性制导回溯技术（dependency directed backtracking）将会很有趣。

惰性展开

在一个本体或描述逻辑 TBox 中，大的复杂的概念很少单个描述，而是从其描述较为简单的指定概念的层级建立。基于 tableau 的算法可通过在添加源于 TBox 的表达式之前试图在概念名称之间寻找矛盾来利用这个结构。这一策略被称为*惰性展开*[10，62]。

惰性展开的好处可通过在词法上*规格化*（normalising）和*命名*（naming）所有的概念表达式以及递归地命名它们的子表达式而最大体现出来。对一个表达式 C 规格化可通过用标准形式（如析取重写成合取的非形式）来重写它；通过用一个新的概念名称 A 替代它来命名，并且把公理 $A \equiv C$ 添加到 TBox 中。规格化步骤允许词法上等价表达式的识别与等价命名，甚至还可以检测语法上“明显的”可满足性和不可满足性。

吸收

并不是所有的公理都服从惰性展开。特别地，所谓的一般概念包含（GCI）,形如 $C \subseteq D$ 的公理（其中 C 是非原子的），必须通过明确地使在模型中的每一个个体成为 $D \cup \neg C$ 的一个实例来处理。大量的 GCI 会导致高度的非确定和灾难性的性能下降[62]。

吸收是另外一种重写技术，它试图通过将 TBox 中的 GCI 吸收入形如 $A \subseteq C$ 的公理来减少它们的数量，其中 A 是一个概念名称。基本思想是，一个形如 $A \cap D \subseteq D'$的公理可以重写成 $A \subseteq D' \cup \neg D$,且被吸收进一个存在的公理 $A \subseteq C$,使得 $A \subseteq C \cap (D' \cup \neg D)$[70]。尽管析取仍然存在，惰性展开保证了它只应用到那些已知是 A 的实例的个体上。

相关性制导回溯

隐藏在子表达式中的内在的不可满足性会导致大量没有结果的回溯搜索称为颠簸（thrashing）。例如，展开表达式$(C_1 \cup D_1) \cap \ldots \cap (C_n \cup D_n) \cap \exists R.(A \cap B) \cap \forall R.\ \neg A$ 会导致在发现$\exists R.(A \cap B) \cap \forall R.\ \neg A$ 的内在的不可满足性之前，$(C_1 \cup D_1) \cap \ldots \cap (C_n \cup D_n)$的 2^n 个

可能扩展的无效探索。这一问题可通过修改一个相关性制导回溯的形式（称为*回跳法*（backjumping））来处理，它已被用来解决约束可满足性问题[20]。

回跳法通过用一个相关集来标示概念以指明它们所依赖的非确定扩展选择而起作用。当发现冲突时，冲突概念的相关集可以用来识别最近的非确定扩展，从中选择一个备选方案可能可以减少冲突的发生。于是，算法可以在没有探测任何一个备选方案的情况下就介入非确定扩展来跳回。类似的技术已用在一阶定理证明器中，如，在 HARP 定理证明器中使用的“证明凝聚（proof condensation）”技术[82]。

1.7 SHIQ 的扩展和变体

如前面所说，本体语言 DAML+OIL 和 OWL 用名词性词和具体数据类型扩展了 SHIQ。在这一节里，我们讨论这些扩展在 SHIQ 的推理问题中的重要性。

具体数据类型，在 DAML+OIL 和 OWL 中均可用，它是具体域的一个非常受限的形式[11]。例如，利用全体非负整数的具体域并配有<谓词，一个把（抽象）个体与他们的（具体）年龄关联起来的（函数化）角色 age，以及 hasParent 的一个(函数化)子角色 father,如下的公理声明孩子们比他们的父亲年轻：

$$\text{Animal} \subseteq (\text{age} < (\text{father} \circ \text{age}))$$

使用具体域扩展富有表达力的描述逻辑很容易导致不可判定性[12，75]。但是，DAML+OIL 和 OWL 只提供具体域非常有限的形式。特别地，具体域一定不允许元数大于 1 的谓词（如在我们例子中的<），并且谓词限制一定不包含角色链（如在我们例子中的 father∘age）。在[84]里展示了用一种稍微更通用的具体域来扩展的 SHIQ 的可判定性。

关于名词性词，事情变得有点复杂了。首先，我们认为可用与在[97]中的 SHIQ 上用的相同的（相对公理化）技术来把用名词性词扩展了的 SHIQ 转成 C2 的段，带有记数量词的一阶逻辑的双变量段。因为这个转换是多项式的，可满足性和包含可在 NEXPTIME 判定。这是最理想的，因为这个问题也是 NEXPTIME-hard 的[97]。粗略地说，GCI（或传递角色和角色层级）、逆角色以及带有名词性词的数字限制的结合对于这个在复杂度上的跳跃（从对于 SHIQ 的 EXPTIME 到 NEXPTIME）是重要的。据我们所知，对于带有名词性词的 SHIQ 目前还没有“可行的”判定程序被描述。“可行的”的意思是一个以某种“目标导引”的方式运作的判定程序，不同于“盲目地”猜测至多指数大小的一个模型 I 然后检测 I 是否确实是输入的一个模型。

1.8 新的推理服务

正如引言中所讨论到的，标准描述逻辑推理服务（如可满足性和包含算法）可用在本体生命周期的不同阶段。在设计阶段，它们可测试概念是否是没有矛盾的以及是否能从概念间得到隐含的关系。但是，对于这些服务的应用，需要一个十分先进的 TBox。推理的结果可以用来进一步地开发 TBox。但是，直到现在，描述逻辑系统对于写这个初始化 TBox 并没有提供推理支持。在描述逻辑中的所谓非标准推理（像计算最小公包含[36，17，71，73]，最专指的概念[14，72]，重写[18]，近似[29]，以及匹配[24，16，15，5]）的开发，试图解决这个不足。接下来，我们将概述这些新的推理如何能支持建立一个描述逻辑知识库。

假设知识工程师想引进一个新概念的定义到 Tbox 中。在许多情况下，她将不会从零开始开发这个新的定义，而是试图重用已经存在于一些知识库（不管是她现在正在建的还是以前建的）中的东西。在一个化学处理工程应用中[89，78]，我们观察到两种在实践中已经实现的方法：

1． 知识工程师在新定义概念的基本结构上作决定，然后试图寻找有类似结构的已经定义好

的概念。可以修改这些概念来获得新概念。

2. 知识工程师不是直接定义新概念，而是首先给出属于将要定义概念的对象的例子，然后试图把这些例子概括到概念定义中。

这两个方法都可以通过上面提到的非标准推理来支持，尽管这种支持还没有被现存的任何描述逻辑系统提供。

第一种方法可通过针对概念描述匹配概念模式来支持。*概念模式*（concept pattern）是一个可能包含代表描述的变量的概念描述。模式 D 在描述 C 上的匹配符σ通过概念描述取代变量，使得结果概念σ(D)等价于 C。例如，假定知识工程师正在寻找关于有一个儿子和一个女儿他们共享某一特征的个体的概念。这个可由模式

$$\exists \text{ hasChild.}(\text{Male} \sqcap X) \sqcap \exists \text{ hasChild.}(\text{Female} \sqcap X)$$

来表达。代换$\sigma = \{X \mapsto \text{Tall}\}$表明这个模式匹配描述$\exists$ hasChild.(Male $\sqcap$ Tall) $\sqcap \exists$ hasChild.(Female $\sqcap$ Tall)。但是，请注意在一些情况下，匹配符的存在并不如此明显。

第二个方法可通过计算最专指概念以及最小公包含器的算法来支持。假设给定例子 Abox 个体 $i_1,\ldots,i_k$。第一步，通过分别计算将这些个体作为其实例的可用的描述逻辑中的最专指（关于包含）概念 $C_1,\ldots,C_k$, 将这些个体概括到概念中。第二步，通过计算 $C_1,\ldots,C_k$ 的最小公包含器，即包含 $C_1,\ldots,C_k$ 的最小概念描述（在可用的描述逻辑中），把这些概念概括到一个概念中。在这个背景下，概念的重写开始活动，因为由用来计算最小公包含器的算法产生的概念描述可能相当大（因而并不容易让知识工程师理解和修改）。通过引进在 Tbox 中定义的名称，重写最小化了这些描述的大小，但并没有改变它们的意思。

迄今，这种非标准推理上的结果还限制在那些表达力大大低于 SHIQ 的描述逻辑中。对于它们中的一部分，它们只在为缺乏表达力的描述逻辑所使用时有意义。例如，在包含析取构造器的描述逻辑中，$C_1,\ldots,C_k$ 的最小公包含器只简单地是它们的析取，并且计算这个对于知识工程师来说并没有任何作用。知识工程师最想获得的最小公包含器的计算结果，是输入概念之间的结构相似性。

因而，只有在使用受限表达能力的描述逻辑时，才可能提供非标准推理的支持。然而，这也只在本体工程的环境下才有意义。事实上，需要最多支持的用户是幼稚的，有理由假定他们将不会用到（或者被提供）底层描述逻辑的全部表达能力。这个二级方法已经呈现在工具中，如 OilEd[21]，它提供了一个框架式的用户界面。利用这个简单的界面，人们可以只使用 OIL 的一段表达能力。要使用全部表达能力，人们必须用描述逻辑表达式。

解决在不同表达能力的描述逻辑之间的差异的另一个方法是利用近似推理[29]。这里，试图通过用一个更小表达能力的描述逻辑 L_2 中的描述 D，来逼近一个富有表达能力的描述逻辑 L_1 中一个给定的概念描述 C。当自上逼近时，D 应该是 L_2 中包含 C 的最小描述，而当自下逼近时，D 应该是 L_2 中被 C 包含的最大描述。

1.9 结语

在描述逻辑研究中关于形式化的基于逻辑的语义的侧重以及基本推理问题的彻底调查研究，连同对于富有表达能力的描述逻辑的高度优选系统的可用性，使这个知识表示形式体系家族成为定义本体语言的理想起始点。支持高质量本体的构建、集成及演化所需的推理服务由为富有表达力的语言而设计的先进的描述逻辑系统提供。

但是，为了能实际应用，这些语言也需要基于描述逻辑的工具，进一步支持知识获取（即，构建本体）、维护（即本体的演化）以及本体的集成和互操作（inter-operation）。在这个方向上的第一批步骤已经完成了。如，OilEd 是支持 OIL，DAML+OIL 和 OWL 本体开发的工具。在一个更基础的级别上，支持建立和维护描述逻辑知识库的非标准推理现在是描述逻辑研究

的一个重要的课题。所有这些努力的目的都是为了支持那些不是描述逻辑专家的用户建立和维护描述逻辑知识库。