知识编译研究

谷文祥 赵晓威 殷明浩
(东北师范大学计算机学院 长春 130117)

摘 要 知识编译是处理命题逻辑知识库常用的一种新技术,知识编译的过程就是把知识转换成易于推理的表示形式。作为一种有效的推理机制,这种技术已经被广泛地应用到各种各样的人工智能领域中。对目前知识编译的研究与应用进行了综述性的介绍,并且从简洁性、多项式时间内支持的查询操作以及多项式时间内支持的转换操作3个方面对各种知识编译目标语言做了系统的阐述。

关键词 知识编译,编译目标语言,命题逻辑,知识库
中图法分类号 TP18 **文献标识码** A

Knowledge Compilation Survey

GU Wen-xiang ZHAO Xiao-wei YIN Ming-hao
(School of Computer Science, Northeast Normal University, Changchun 130117, China)

Abstract Knowledge compilation (KC) has been emerging as a new technology for dealing with the propositional logic database. The process of it is transforming the given knowledge into another form, and then the reasoning can be done more tractable on it. As an effective reasoning method, KC has been applied in various artificial intelligence areas. We introduced the details about the research and application of KC, and proposed a perspective on target compilation languages according to the succinctness, the class of queries and transformations that the language supports in polytime.

Keywords Knowledge compilation, Target compilation languages, Proposition logic, Knowledge base

1 引言

知识编译(Knowledge Compilation)是20世纪90年代出现的一个研究方向[1]。最初,这种技术用来处理通用命题推理的难求解性,但经过十几年的发展,它已经被广泛地应用到人工智能的许多领域中。通常,大规模推理问题的效率都很低。为了解决这个问题,人们提出了很多方法,知识编译就是其中的一种。知识编译的实质就是对知识的处理及转换,它把推理问题分成两个阶段:第一个阶段叫做离线推理(off-line reasoning)阶段,在这个阶段中知识库 Σ 被处理,转换成易于推理的表示形式;第二个阶段叫做在线推理(on-line reasoning)阶段,在这个阶段中当询问(query) C 实际到来的时候,再利用第一阶段处理的结果回答 C 是否能被 Σ 逻辑推出。这样,推理的效率会得到提高。而且重要的是,对于不同的询问,知识编译只需要做一次。用户关心的是第二阶段的时间,至于知识编译过程所花的时间可以通过对同一个知识库的多次询问得到补偿。

知识编译与推理的关系类似于排序和查找的关系:假设有一个数据库,用户需要对其进行查找操作。当然,对于原始的、未经过任何组织的数据也可以进行查找,但效率会比较低。如果对它进行适当的组织、排序,查找的效率会有所提高。知识编译与这个过程类似。当然,知识编译的过程不是简单地对知识进行排序的过程,而是把知识转换成易于推理的表示形式的过程。我们可以从3个方面评价各种不同的知识编译方法:编译目标语言的简洁性;编译目标语言支持的多项式时间内查询操作;编译目标语言支持的多项式时间内转换操作。

目前,知识编译得到大量研究人员的广泛关注,一方面是由于这种技术尽可能地将大量的计算消耗放在离线阶段,并且仅处理一次,而在在线阶段,指数的高效查询就会抵消这些消耗,如它在诊断系统中的应用;另一方面,利用知识编译得到编译目标语言及相应的算法都很简单,便于我们为软硬件平台开发在线推理系统,如它在电子消费中的应用。

本文第1节简单地回顾知识编译方法;第2节介绍知识编译方法的分类;第3节介绍、评价各种编译目标语言的标准,并给出目前有代表性的编译目标语言的定义及性质;第4节介绍知识编译在模型诊断和规划领域中的应用;第5节给出知识编译图谱;最后总结全文并指出未来研究工作的方向。

2 知识编译的种类

对于知识编译技术,研究人员感兴趣的是在线阶段编译目标语言具有的性质。因此本文不介绍具体的知识编译方

到稿日期:2009-08-24 返修日期:2009-10-25 本文受国家自然科学基金项目(编号为:60473042,60573067和60803102)资助。
谷文祥(1947—),男,教授,博士生导师,主要研究方向为智能规划与规划识别、形式语言与自动机理论、模糊数学及其应用,E-mail:gwx@nenu.edu.cn;赵晓威(1984—),女,硕士生,主要研究方向为智能规划与规划识别;殷明浩(1979—),男,讲师,主要研究方向为智能规划与规划识别、自动推理,E-mail:ymh@nenu.edu.cn(通信作者)。

法，而把阐述的重点放在对各种编译目标语言性质的介绍上。首先从两个角度对知识编译方法进行分类：

2.1 根据知识编译方法的处理对象

1)命题逻辑的知识编译

在过去的几十年中，命题逻辑在各种应用领域中被广泛使用，如知识表示。而知识编译技术是处理知识的一种有效方法：我们先用命题逻辑来表示知识并建立知识库，然后对知识库进行知识编译，从而得到可以快速响应查询的范式。

目前的各种知识编译方法主要处理命题逻辑表示的支持库，并且已经涌现出十几种编译目标语言，如 OBDD，DNNF，EPCCL 等。这种现象出现的原因在于命题逻辑只包含有 $\wedge$，$\vee$，$\neg$等连接符，因此知识编译过程相对比较简单，而且也可以把知识编译看作是一个 DPLL 的求解过程[23]。近几年，命题逻辑中的知识编译方法已被用于一些表达能力更强的逻辑中去，如下面要介绍的 3 种逻辑。

2)描述逻辑的知识编译

描述逻辑是一阶谓词逻辑的一个可判定子集。它与一阶谓词逻辑不同的是，描述逻辑能提供可判定的推理服务，它是当前语义网发展中本体的理论基础。由于描述逻辑中引入了存在量词、全称量词等一些算子，因此对其进行知识编译就需要事先选择一个适合的目标语言。

Furbach[25]首次将知识编译技术用于描述逻辑(所选目标语言为 Full Dissolvent)，对编译后的目标语言，概念之间的可满足性和蕴含查询能够在多项式时间内完成，开创了这个领域研究的先河。随后，Bienvenu[26]提出将 Prime Implicate 范式作为描述逻辑的目标语言，进一步发展了这种策略，同时表明了知识编译技术可以显著提高描述逻辑的推理能力。由于描述逻辑在语义 Web 研究上有重要作用，因此研究描述逻辑中的知识编译，不但具有一定的学术价值，而且具有较高的应用价值。

3)可能性逻辑的知识编译

在人工智能领域中，可能性逻辑提供了一个简单且严格的方法来处理自动推理过程中遇到的不确定和有优先次序的不完备知识。它的一个重要特征就是能够处理不一致的知识库：可能性逻辑公式[51]是由经典逻辑公式 p 和度 a 组成的二元组(p,a)，$a\in(0,1]$，度 a 表示确定或优先程度。

对可能性知识库进行知识编译后，就可以在多项式时间内对知识库进行可满足和蕴含查询。目前，仅有 Benferhat[27]成功地将 DNF 作为可能性逻辑的编译目标语言。我们可以沿着这一思路，尝试将其它语言作为可能性逻辑的编译目标语言，提高可能性逻辑的推理效率。

4)一阶逻辑的知识编译

一阶逻辑限定了对象和关系的存在，比命题逻辑具有更强的表达能力，然而，对一阶逻辑进行知识编译，通常会面临两个问题：编译算法的终止、编译理论的有限性和推理的半可判定性。尽管如此，由于一阶逻辑本身的表达能力强，我们不能忽视对其进行知识编译的重要意义。

Alvaro 成功地将命题逻辑中近似知识编译方法应用于一阶逻辑，分别使用 GLB(Greatest Lower Bound)[50]和 LUB(Lowest Upper Bound)[49]两种近似编译方法，能够同时解决一阶逻辑知识编译过程中遇到的问题。满足限定条件的语言都可以作为其编译的目标语言。

2.2 根据知识编译方法的完备性和可靠性

1)精确的知识编译

精确的知识编译方法就是将知识库中的知识编译为等价的目标语言形式的知识。由于目标语言需要满足某些性质，因此知识编译技术通常会增加知识库的规模。为了准确地回答对知识库的查询，大多数知识编译方法都是精确的。通常，我们会在知识编译过程中加入一些启发式，来减小目标语言的规模。以 PI(prime implicates)为例，采用精确知识编译后得到的知识库等价于它的所有 PI 范式的合取[28]。

2)近似的知识编译

近似的知识编译方法能够减少编译过程的计算复杂性，但是它是以牺牲蕴含查询时的可靠性和完备性为代价的。也就是说，响应一个查询所使用的编译知识库可以是不合理或是不完备的。许多实例证明，并不是所有的在线推理都是高效的，因此可以采用近似的知识编译方法：语言限制和理论近似。这两种方法经常被用于实现易处理的推理问题，其特点是在线推理和离线推理效率上没有什么区别。

3 知识编译的目标语言

在介绍各种编译目标语言之前，首先给出它们的性质。

3.1 相关性质

在这一节中将详细介绍已有的十几种编译目标语言。对于命题逻辑的基本概念，如可满足、蕴含($\models$)和等价($\equiv$)等，由于篇幅限制，就不再介绍了。本文中提到的所有语言都是 NNF 语言的子集，并且任意命题语句都可以表示为 NNF 中的一个句子，它的定义如下[2]。

定义 1 令 PS 是命题变量集合，NNF_{PS} 中的句子是一个带根的、有向非循环图(DAG)，如图 1 所示，其中叶结点被标记为 true，false，X 或$\neg X$，且有 $X\in PS$；每一个内部结点被标记为 $\wedge$ 或 $\vee$，并且可以有任意多个子女。NNF 中的句子 Σ 的长度表示为$|\Sigma|$，表示对应 DAG 图中边的个数，高度指的是对应 DAG 图中从根到叶结点最大的边数。

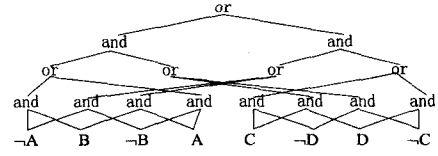

图 1 NNF 中子句的 DAG 表示

编译目标语言和常见的知识表示语言是不同的。知识表示语言是人们在通常情况下进行读和写的语言，例如 CNF 就是一种广泛使用的知识表示语言，以及 Horn 子句；而编译目标语言不需要满足人们读和写的要求，但是它必须能够满足多项式时间内的查询/转换操作。对知识表示语言的介绍不在本文的论述范围之内。我们称一个语言是编译目标语言，它还需要满足多项式时间内的子句蕴含测试。很显然，除非 P=NP[3]，否则 NNF 语言不满足这一要求，但是它的许多子集却可以被看作是编译目标语言。

不同的编译目标语言具有不同的性质，下面介绍比较重要的一些性质。从现在开始，我们约定如果 C 是 NNF 某一子集中的结点，那么 $Vars(C)$ 表示以结点 C 为根的变量集合。

扁平性(Flatness)：DAG 中句子的高度最大是 2。

简单析取(Simple-disjunction)：DAG 中每一个或结点(标

记为∨)的后代都是叶结点,并且它们不共享变量。

简单合取(Simple-conjunction):DAG 中每一个与结点(标记为∧)的后代都是叶结点,并且它们不共享变量。

分解性(Decomposability):NNF 语言的一个子集 L 满足这一性质,当且仅当对于任意一个合取式 C,C 的合取项不共享变量。也就是说,如果 $C_1,C_2,\cdots,C_n$ 是与结点 C 的后代,那么当 $i\neq j$ 时,有 $Vars(C_i)\cap Vars(C_j)=\Phi$ 成立。

结构化分解性(Structured Decomposability):NNF 语言的一个子集 L 满足这一性质,当且仅当 L 与一个 v-tree[4] T 相对应,对于任意的合取项 α 和 β,存在 T 中的一个结点 t,使得 $Vars(\alpha)\subseteq Vars(t_l)$ 和 $Vars(\beta)\subseteq Vars(t_r)$ 同时成立。从本质上讲,具有这一性质的 L 能够满足 $\wedge BC$ 转换操作。

确定性(Determinism):NNF 语言的一个子集 L 满足这一性质,当且仅当对于任意一个析取式 C,C 的任意两个析取项是逻辑矛盾的。也就是说,如果 $C_1,C_2,\cdots,C_n$ 是或结点 C 的后代,那么当 $i\neq j$ 时,有 $C_i\wedge C_j\models$ false 成立。

平滑性(Smoothness):NNF 语言的一个子集 L 满足这一性质,当且仅当对于任意一个析取式 C,C 的任意两个析取项含有相同的变量。也就是说,如果 $C_1,C_2,\cdots,C_n$ 是或结点 C 的后代,那么当 $i\neq j$ 时,有 $Vars(C_i)=Vars(C_j)$ 成立。

决策性(Decision):NNF 语言的一个子集 L 中决策结点满足这一性质,当且仅当这个结点被标记为 true,false,或者它是一个 $(X\wedge\alpha)\vee(\neg X\wedge\beta)$ 形式的或结点,其中 X 是变量,α 和 β 是决策结点。

有序性(Ordering):NNF 语言的一个子集 L 满足这一性质,当且仅当集合 PS 中的变量是有序的(<)。

3.2 评价标准

在这一节中将介绍各种编译目标语言的 3 个评价标准:简洁性、多项式时间内支持的查询/转换操作。通过本小节的介绍,希望能够加深读者对编译目标语言的理解。

3.2.1 简洁性

目前已经出现了几十种编译目标语言,它们都是 NNF 语言的子集。其中一些语言是我们熟知的,如 OBDD,PI 和 DNF,并且已经能够清楚地了解它们的性质。其它一些语言,如 EPCCL,FNNF 和 Ri-trie 等,是近几年涌现出来的,我们还不能确定它们的某些性质。下面给出简洁性的定义。

定义 2[5] 令 L_1 和 L_2 是 NNF 语言的两个子集,我们说 L_1 至少和 L_2 一样简洁($L_1\leqslant L_2$),当且仅当存在多项式时间 P,对于任意的句子 $\alpha\in L_2$,都存在一个等价的句子 $\beta\in L_1$,使得 $|\beta|\leqslant_p|\alpha|$ 成立。其中,$|\alpha|$ 和 $|\beta|$ 分别表示 α 和 β 的大小。

需要指出的是,对于句子 α,不要求存在一个算法能在多项式时间内输出 β,仅要求证明 β 的存在性。关系≤具有传递性,即前序关系,类似地也可以定义<,那么 $L_1<L_2$ 当且仅当 $L_1\leqslant L_2$ 成立,$L_2\leqslant L_1$ 不成立。

3.2.2 多项式时间内的查询操作

对于某一应用领域,在判断一个编译目标语言是否适用时,应当不仅仅考虑它的简洁性,同时也要考虑这种语言在多项式时间内支持的查询和转换操作。接下来介绍几种查询操作:一致性(CO)、有效性(VA)、子句蕴含(CE)、蕴含库(IM)、库等价(EQ)、库蕴含(SE)、模型计数(CT)和模型枚举(ME)。

L 满足 CO(VA)查询,当且仅当如果 Σ 是可满足的(有效的),存在多项式时间的算法能将 L 中的任意公式 Σ 映射到 1 上;否则映射到 0 上。

L 满足 CE 查询,当且仅当如果 $\Sigma\models\gamma$ 成立,存在多项式时间的算法能将 L 中的任意公式 Σ 和 NNF 中的任意子句 γ 映射到 1 上;否则映射到 0 上。

L 满足 IM 查询,当且仅当如果 $\gamma\models\Sigma$ 成立,存在多项式时间的算法能将 L 中的任意公式 Σ 和 NNF 中的任意项 γ 映射到 1 上;否则映射到 0 上。

L 满足 EQ(SE)查询,当且仅当如果 $\Sigma\models\Phi$ ($\Sigma\equiv\Phi$)成立,存在多项式时间的算法能将 L 中的任意公式 Σ 和 Φ 映射到 1 上;否则映射到 0 上。

L 满足 CT 查询,当且仅当存在多项式时间的算法能将 L 中的任意公式 Σ 映射到一个非负整数上,而且这个整数表示 Σ 中模型的数目。

L 满足 ME 查询,当且仅当在多项式时间 $p(.,.)$ 内,存在一个算法能够在 $p(n,m)$ 时间内输出 L 中任意公式 Σ 的全部模型,其中 n 是 Σ 的大小,m 是 Σ 的模型数目。

3.2.3 多项式时间内的转换操作

令 L 表示 NNF 语言的某一子集,下面介绍与编译目标语言有关的几种转换操作:调节(CD)、忘记变量(FO)、忘记单个变量(SFO)、合取(∧C)、析取(∨C)、有界合取(∧C)、有界析取(∨C)和否定(¬C),转换操作的结果会返回 L 的其它表示形式。

L 满足 CD 转换,当且仅当存在多项式时间的算法能将 L 中的任意公式 Σ 和可满足的项 γ 映射到 L 中的一个公式上,而且这个公式与 $\Sigma|\gamma$ 逻辑等价。

L 满足 FO(SFO)操作,当且仅当存在多项式时间的算法能将 L 中的任意公式 Σ 和 PS 中变量的子集 X 映射到 L 中的一个公式上,而且这个公式与 $\exists X.\Sigma$ 逻辑等价。如果这个性质仅对含有一个变量的集合 X 成立,我们就说 L 满足忘记单个变量操作。

L 满足 ∧C(∨C)转换,当且仅当存在多项式时间算法能将 L 中任意的有限公式 $\Sigma_1,\cdots,\Sigma_m$ 集合映射到 L 中的一个公式上,并且这个公式与 $\Sigma_1\wedge\cdots\wedge\Sigma_m$ ($\Sigma_1\vee\cdots\vee\Sigma_m$)逻辑等价。

L 满足 ∧BC(∨BC)转换,当且仅当存在多项式时间的算法能将 L 中的任意两个公式 Σ 和 Φ 映射到 L 中的一个公式上,而且这个公式与 $\Sigma\wedge\Phi$ ($\Sigma\vee\Phi$)逻辑等价。

L 满足¬C 转换,当且仅当存在多项式时间的算法能将 L 中的公式 Σ 映射到 L 中的一个公式上,而且这个公式与 $\neg\Sigma$ 逻辑等价。

3.3 编译目标语言

区分各种知识编译方法的关键在于我们要将给定的知识库编译为哪种目标语言。下面详细介绍目前流行和常用的一些编译目标语言。了解编译目标语言的性质后,我们容易理解,f-NNF 是满足扁平性质的 NNF 语言的子集,s-NNF 是满足平滑性质的 NNF 语言的子集,d-NNF 是满足确定性质的 NNF 语言的子集,CNF 是满足简单析取性质的 NNF 语言的子集,而 DNF 是满足简单合取的 NNF 语言的子集。其中,CNF 是最常见的一种语言,但是由于 CNF 不能够满足多项式时间内的子句蕴含测试,因此 CNF 不能作为一种编译目标语言(除非 P=NP),但是 DNF 却可以满足这一性质。

3.3.1 HORN-C,KROM-C 和 AFF

HORN-C,KROM-C,AFF,K/H-C 和 renH-C 都是 CNF

的子集，这几种编译目标语言都是不完备的，即并不是所有的命题语句都可以用它们之一完全表示。例如，不存在它们中的任意一种公式表示能够与 CNF 公式 $(a \vee b \vee c) \wedge (\neg a \vee \neg b \vee \neg c)$ 等价，然而建立在这 5 种语言基础上的 KROM-C[∨]，HORN[∨]，K/H-C[∨]，renH-C[∨]和 AFF[∨]却是完备的编译目标语言[9]。

HORN-C 语言中的每一个子句都至多包含一个正文字；KROM-C 语言中每一个子句都是二元的；K/H-C 语言是 HORN-C 和 KROM-C 语言的并集；对于任意 CNF 公式 α，在 renH-C 中存在 Var(α)的一个子集 V(称为 α 的 Horn 重命名)，这时将 α 中所有出现在 V 中的文字 l 用它的互补文字 $\neg l$ 替换，得到的公式是一个 HORN-C 公式；AFF 语言是由 XOR 子句合取构成的；KROM-C[∨]，HORN[∨]，K/H-C[∨]，renH-C[∨]和 AFF[∨]语言分别是由对应公式析取构成的。

例如：

$(a \vee b) \wedge (\neg b \vee c)$是 KROM-C 公式；

$(\neg a \vee \neg b \vee c) \wedge (\neg b \vee c \vee \neg d)$是 HORN-C 公式；

$(a \vee b \vee c) \wedge (\neg a \vee \neg b \vee c)$是 renH-C 公式；

$(a \oplus \neg b) \wedge (\neg a \oplus c)$是 AFF 公式；

$((\neg a \vee c) \wedge (\neg b \vee c \vee \neg d)) \vee (a \wedge b)$是 HORN-C[∨]公式。

AFF[∨]和 KROM-C[∨]比 DNF 更简洁，并且支持多项式时间内的相同查询/转换操作，在实际应用中优于 DNF[29]；KROM-C[∨]，HORN[∨]，K/H-C[∨]，renH-C[∨]和 AFF[∨]比 IP 更简洁，但是多项式时间内支持的查询操作要少于 PI；当不考虑完备性和转换操作时，renH-C 和 AFF 是进行知识编译时最好的选择，因为这两种语言的表达能力在 5 种不完备语言中较强，并且支持多种多项式时间内的查询操作。

3.3.2 PI 和 IP

通常，公式 Σ 蕴含的最小子句被称为是 Σ 的本源蕴含(PI)，蕴含公式 Σ 的最少文字合取构成的项被称为 Σ 的本源蕴含式(IP)。蕴含在非单调推理中的一些方法上很有用[30-32]，这时需要计算出公式的所有蕴含；另外，在硬件验证的故障诊断中，往往需要事先得到公式所有的模型，这时蕴含式就体现出它的特性。目前，已经有许多种方法可用来计算给定命题公式的 PI(IP)[33-40]。它们的严格定义为：

定义 3[6] PI 是 CNF 语言的子集，每一个被句子 Σ 蕴含的子句 C，都能够推出 Σ 中的任意一个子句，并且 Σ 中没有任何两个子句可以相互推出。

作为 PI 的对偶语言 IP，也可以按照同样的方式定义。

定义 4[6] IP 是 DNF 语言的子集，每一个蕴含句子 Σ 的项 T，都能够被 Σ 中的任意一个项推出，并且 Σ 中没有任何两个项可以相互推出。

自从 Reiter[41]提出这种知识编译方法以来，许多研究者将工作集中在如何使 PI 的表示更加简洁，进而使多个 PI 能在多项式空间内被表示出来，例如 Trie 表示法[7]、ZBDD 表示法[8]。但是这些表示方法与 PI 有着质的不同，它们不需要支持与 PI 相同的查询和转换操作并且在简洁性上也有很大的不同。

PI 和 IP 能够支持多项式时间内除 CT 以外的全部查询操作，以及 CD，FO 和 ∨BC 等转换操作。

3.3.3 DNNF 及其子集

定义 5[10] DNNF 是满足可分解性质的 NNF 语言的子集；d-DNNF 是满足可分解性和确定性的 NNF 语言的子集；sd-DNNF 是满足可分解性、确定性和平滑性的 NNF 语言的子集。

图 1 就是一个 DNNF 公式的 DAG 表示。容易发现，每一个 and 结点的子树之间都不含有相同原子。图 1 也是一个 d-DNNF 公式的 DAG 表示，每一个 or 结点的任意两个子树表示的公式可以推出矛盾。图 1 还是一个 sd-DNNF 公式的 DAG 表示，每一个 or 结点的任意两个子树都含有相同的原子。

DNNF 这种语言的特点是它支持大量的多项式时间内逻辑操作，而这些操作可以有效地应用于人工智能各种领域中。如在基于模型的诊断系统中，已经成功地利用编译后的 DNNF 范式快速回答各种各样的诊断查询，并且这种范式的空间复杂度很低[45]。目前已经有人开发出将 CNF 编译为 d-DNNF 的编译器 c2d[44]，它是目前已有的编译器中效率最高的。

定义 6[11] $\mathrm{DNNF_T}$ 是满足结构化分解性质的 DNNF 的子集；$\mathrm{d\text{-}DNNF_T}$ 是满足结构化分解性质的 d-DNNF 的子集。

与 DNNF 和 d-DNNF 相比较，$\mathrm{DNNF_T}$ 和 $\mathrm{d\text{-}DNNF_T}$ 能够支持多项式时间内的有边界合取操作(∧BC)。利用这一性质，可以对 $\mathrm{DNNF_T}$ 和 $\mathrm{d\text{-}DNNF_T}$ 范式构成的知识库进行增量的知识编译[12]，从而扩大了这种语言的应用范围，如模型检测。

DNNF 能够支持多项式时间内的 CO，CE，ME，FO 和 ∨C 操作，d-DNNF 和 sd-DNNF 能够支持多项式时间内的 CO，VA，CE，IM，CT，ME 和 CD 操作。需要强调的是，给 d-DNNF 语言加上平滑性约束，并没有影响它自身的结构属性。d-DNNF 和 sd-DNNF 的简洁性相同，并满足相同的操作。

DNNF 是目前应用最广泛的编译目标语言，被成功地应用于基于模型的诊断系统中，这种语言的特性亦使得它能够被用于复杂的智能规划领域。并且，它是仅有的几个能满足 FO 的语言之一，FO 操作能够带来的好处有：

(1) 如果预先知道某些原子不会出现在对知识库的查询里，那么就利用 FO 操作将它们从知识库中“编译”出去。也就是说，假设集合 P 包含 DNNF 句子 Δ 中的所有原子，而集合 X 包含集合 P 不会出现在查询中的原子，那么就可以在 Δ 上进行 FO 变量集合 X 的操作，得到的结果仍然能有效地回答对知识库的查询。

(2) 在某些应用领域中，通常要获取关于一些原子的描述信息，例如在规划和诊断领域里，在规划时这些原子用来表示动作流；在诊断中，这些原子用来表示缺省变量。这样利用 FO 操作，就可以将那些不感兴趣的原子“编译”出去。

(3) 在电路表示上，用某种编译目标语言 Δ 描述某一电路时，可以用 FO 操作消除 Δ 的中间变量。

3.3.4 BDD 及其子集

定义 7[11] BDD 是 NNF 语言的子集，其中每个句子的根结点都是决策结点。有时它也以二元决策图的形式给出，即含有一个根结点和两个叶结点 0 和 1 的无向循环图，如图 2 所示；FBDD 是 DNNF 和 BDD 的交集；OBDD< 是满足有序

性质的 FBDD 的子集；OBDD 是所有 OBDD< 的并集。

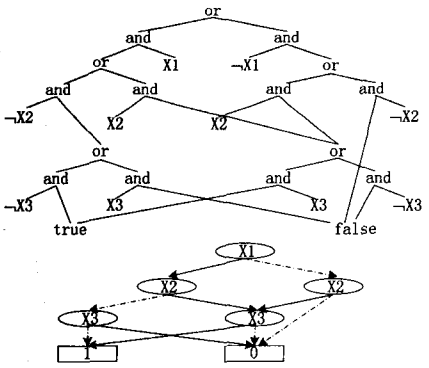

图 2　上边是 BDD 中子句 DAG 表示，下边是对应的二元决策图

BDD 不能作为一个编译目标语言(除非 P = NP)，但是它的几个子集却是编译目标语言。另外，BDD 的子集与 DNNF 有着密切的关系：OBDD 可以在线性时间内转化为 DNNF；FBDD 的编译算法可以立即成为 DNNF 的编译算法。也就是说，DNNF 编译算法的稳定性更好。OBDD 在验证领域有着广泛的应用，常被用于判断一个布尔型函数和它的电路实现之间的等价性。

OBDD 支持多项式时间内除 SE 以外的所有查询操作，支持 CD，SFO 和 ¬C 3 个转换操作；OBDD< 能够支持多项式时间内的全部查询操作和除 FO，∧C 和 ∨C 以外的转换操作。

3.3.5　MODS

定义 8　MODS 是满足确定性和平滑性的 DNF 的子集。

如图 3 所示，MODS 是 NNF 语言的子集中最易处理的语言之一，可以很容易地找出公式的全部模型。MODS 能够支持多项式时间内全部的查询操作和除 ∧C，∨C，∨BC 和 ¬C 以外的转换操作。由于这种语言表示的简洁性最差，因此它在实际中使用得很少，通常用来与其它语言做比较。

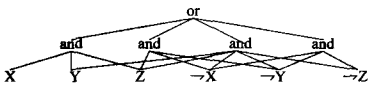

图 3　MODS 语言中的句子

前面介绍的编译目标语言是出现比较早的语言，它们的性质已经被深入地研究。简洁性的比较结果是：

DNNF<d-DNNF<FBDD<OBDD<OBDD$_<$<MODS

除了 PI 以外，DNNF 是最简洁的编译目标语言。我们能确定 PI 没有 DNNF 简洁，但是还不清楚 DNNF 是否比 PI 简洁。现在知道，DNNF 比 OBDD 简洁，那么在验证领域中为什么还要使用 OBDD 呢？那是由于 DNF 是 DNNF 的子集，而测试两个 DNF 公式的等价性是一个 co-NP 完备问题，因此不能在多项式时间内测试两个 DNNF 的等价性，DNNF 在保证空间简洁性的同时也牺牲了其它的性质。

3.3.6　EPCCL

前面介绍的编译目标语言都是通过归结方法得到的。林海提出了一种与归结互补的知识编译方法——扩展规则[12]，并得到一种新的编译目标语言 EPCCL。扩展规则方法不同于已有的知识编译方法，它的主要思想是沿着归结的逆向进行扩展并且用包含排斥原理解决这样做带来的空间复杂性问题。对于 EPCCL 语言，无论是在线推理阶段还是离线推理阶段都是基于扩展规则的，这种方法有时在效率上要优于基于归结的方法。

定义 9　给定一个子句 C 和一个集合 M：

$D = \{C \vee a, C \vee \neg a \mid a$ 是一个原子，$a \in M$ 和 $\neg a$ 都不在 C 中出现$\}$

我们把从 C 到 D 中元素的推导过程叫做扩展规则，D 中的元素叫做应用扩展规则的结果。已经知道，子句 C 和它扩展后的结果是等价的，这样就能够将扩展规则作为一种知识编译方法。

例如，$\Sigma = \{A \vee B \vee D, B \vee C\}$，用扩展规则对子句 $A \vee B \vee D$ 在 C 上进行扩展，得到 $A \vee B \vee C \vee D$ 和 $A \vee B \vee \neg C \vee D$，其中第一个子句被 $B \vee C$ 蕴含，因此可以删除，得到的子句集是 $\{A \vee B \vee \neg C \vee D, B \vee C\}$。

定义 10　EPCCL 是 CNF 的子集，并且集合中的任意两个子句都包含互补文字。

由于目前尚没有对 EPCCL 语言进行深入研究，它除了支持多项式时间内 CO，CE 和 CT 以外，我们不清楚它是否支持其它的查询和转换操作。实验结果已表明，EPCCL 和 PI 这两种语言是以两种不同的方式包含给定公式的所有蕴含，因而各自呈现出不同的特性。

值得指出的是每种现有的编译方法都有各自的优缺点。而扩展规则方法在有些情况下的运行结果比其它方法的结果呈指数级地小。考虑下面的情况：假设待编译的子句集本身就是一个 EPCCL 语言，那么用扩展规则方法编译后的结果就是它本身。如果编译为其它目标语言，结果可能会是指数级的。

3.3.7　Ri-trie

知识编译技术的目标是快速响应对知识库的查询。一般地，查询响应时间是线性的 $O(n)$，n 为编译后知识库的规模。考虑这种情况，若编译后的知识库的规模是原有知识库规模的指数倍，那么查询响应时间会相当长。发现这一问题后，研究人员立刻把工作集中在如何减小编译后知识库的规模上，如对给定的知识库添加限制或者进行近似编译。Ri-trie[42] 这种编译目标语言的提出，使得我们可以从另一个角度来解决查询时间长的问题。

定义 11　trie[43] 是一种特殊的树形结构，其中的每个分枝表示降序排列的符号序列，常被用来保存大量的字符串。

trie 通常被用来表示字典。这种表示的好处是：字典中的每个文字可以被表示为 trie 中的一个分枝。当需要判断一个字符串是否属于某个字典时，只需遍历 trie 中的相应分枝即可。这个过程要么失败，要么可以在线性时间 $O(m)$ 内完成，m 是字符串的长度。trie 也可以表示逻辑公式，如 PI。

由于 trie 中存在大量的冗余，对其应用 SR1，SR2 和 SR4 3 个简化规则后，就得到 reduced implicate trie (Ri-trie)。例如，假设知识库 D 包含子句 $p \vee q \vee \neg s$，$p \vee q \vee r$，$p \vee r \vee s$ 和 $p \vee q$，变量的顺序为 $p > q > r > s$，那么 D 对应的 Ri-trie 如图 4 所示。

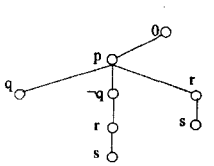

图 4　Ri-trie 的树形表示

对编译为 Ri-trie 的知识库进行查询，可以直接在树上进行遍历操作。响应时间是线性的 $O(n)$，n 为查询的规模，与

知识库的大小无关。这一特点是 Ri-trie 区别于其它编译目标语言的主要特性。

4 知识编译的应用举例

4.1 基于模型的诊断

基于模型的诊断(Model-Based Diagnosis)的主要思想是:根据系统的组成元件与元件之间的连接建立起等待诊断系统模型,这种模型可以用命题逻辑或一阶逻辑语句来描述[20]。为了诊断一个系统,我们可以先建立一个系统模型,然后把它输入到一个诊断设备中去,这个诊断设备处理输入并响应各种各样的诊断查询。基于模型的诊断最初的研究主要集中在模型设备、诊断查询的语义和回答查询的算法上。

而近几年来,基于模型的诊断领域出现了一个新的研究方向,它更多地关注诊断的过程,不是直接在系统模型上进行推理并回答诊断查询,而是将模型转换为另一种表示形式,在这种表示形式上推理并回答诊断查询。这种方法的思想是将计算工作分为两个部分:离线阶段和在线阶段。离线阶段,利用诊断编译器生成设备的编译后模型表示;在线阶段,利用诊断评价器来回答基于编译后模型表示的多次诊断查询[10]。

知识编译技术用在基于模型诊断中的益处:增强了在线诊断系统的效率,把大部分的计算工作放在离线阶段;节省了在线诊断系统所需要的软硬件资源;简化了在线诊断系统,仅需要很少的软硬件就可以将它嵌入到现有的平台中去。

4.2 智能规划

智能规划(Intelligent Planning)是人工智能的重要研究领域之一。智能规划问题是指给定初始状态和目标条件,agent 自动寻找一条从初始状态出发满足目标条件的动作序列[21]。智能规划的主要方法有图规划、基于启发式搜索的规划、基于逐步细化的分层规划、基于模型检测的规划以及基于约束可满足的规划方法。许多研究人员试图将知识编译技术与基于约束可满足的规划方法(SAT)相结合,并且取得了很大的成功[22,24]。基于约束可满足的规划方法的过程如图 5 所示。

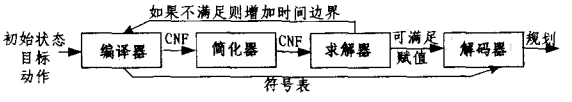

图 5 SAT 规划系统体系结构

SAT 规划系统的体系结构中,编译器(Compiler)的输入是规划问题(包括初始状态、目标状态和动作集合),它首先猜测规划解的长度,产生一个逻辑命题公式。如果逻辑命题公式是可满足的,它蕴涵着规划解是存在的;符号表(Symbol Table)记录了命题变量和规划实例之间的对应;简化器(Simplifier)运用较快(通常为线性时间)的技术(如单元子句法、纯文字消去法等)来降低 CNF 公式的规模;求解器(Solver)运用系统或统计的方法来找到一个满足的赋值;解码器(Decoder)用符号表把赋值转换成一个规划解。如果求解器发现公式是不可满足的,那么编译器就会产生新的编码(代表更长的规划解)。

由基于 SAT 的规划的体系结构可以看到,这种规划方法关键在于两个环节:一个是编码方式,另一个是求解方式。在编码的过程中,若将得到的逻辑命题公式进一步编译为某种目标语言,如 d-DNNF,便可以迅速计算出偏序规划的上界。这样,求解器可以利用深度优先的分枝-定界算法对规划进行剪枝,较早地发现并剪掉不符合要求的规划解,提高规划解生成的时间和空间效率。

将智能规划问题和知识编译方法相结合,是较新的研究方向。相信在将来会有更多的规划研究人员把注意力集中在这个方向上,以加快规划问题的求解效率。

5 知识编译图谱

图 6 给出知识编译图谱,它表示各种编译目标语言之间的关系。

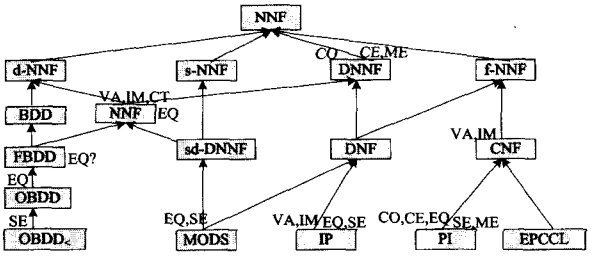

图 6 各种编译目标语言的关系

图 6 中 $L_1 \rightarrow L_2$ 表示 L_1 是 L_2 的子集。在每种语言旁边,列出所有该语言的子集满足而它的超集不满足的查询操作。图 6 中没有 D-FNNF 这种语言,因为我们尚不明确它们与其它语言的关系。

这个图便于系统设计者为实际应用选择适合的编译目标语言:首先明确这一应用需要目标语言满足哪些查询、转换操作,然后设计者选择能支持那些操作最简洁的目标语言。

结束语 本文总结现有的编译目标语言,详细介绍了各种编译目标语言的性质,评价了不同的编译目标语言的标准以及知识编译技术目前的应用和研究。希望对各种编译目标语言的分析,能对相关研究人员有所帮助。在现有语言的基础上提出了新的编译目标语言,它在保证简洁性的同时能满足更多的查询和转换操作。知识编译技术还存在巨大的应用潜力,仍然有许多开发和研究工作需要开展。

参 考 文 献

[1] Selman B, Kautz H. Knowledge compilation and theory approximation [J]. Journal of the Association for Computing Machinery, 1996, 43: 193-224

[2] Darwiche A, Marquis P. A knowledge compilation map [J]. JAIR, 2002, 17: 229-264

[3] Papadimitriou C. Computational complexity[M]. Addison -Wesley, 1994: 154-160

[4] Pipatsrisawat K, Darwiche A. New Compilation Languages Based on Structured Decomposability[C]// Proceedings of AAAI08. 2008: 517-522

[5] Murray N V, Rosenthal E. Efficient Query Processing with Reduced Implicate Tries [J]. Journal of Auto Reasoning, 2007, 38 (1-3): 155-172

[6] Marquis P. Knowledge compilation using theory prime implicates[C]// Proceedings of IJCAI95. 1995: 837-843

[7] Kleer D J. An improved incremental algorithm for generating prime implicates[C]// Proceedings of the 10th National Conference on Artificial Intelligence. 1992: 780-785

[8] Simon L, DelVal A. Efficient consequence finding [C]// Procee-

dings of the 17th International Joint Conference on Artificial Intelligence. 2001:359-365

[9] Fargier H, Marquis P. Extending the Knowledge Compilation Map: Krom, Horn, Affine and Beyond [C] // Proceedings of AAAI08. 2008:442-447

[10] Darwiche A. Decomposable negation normal form [J]. Journal of ACM, 2001, 48(4):608-647

[11] Salem B, Henri P. Compiling Possibilistic Knowledge Bases[C]// Proceedings of ECAI 2006. 2006:337-341

[12] Goran G, Christos H P, Martha S. Incremental Recompilation of Knowledge[C]// Proceedings of AAAI 94. 1994:922-927

[13] Bryant R E. Symbolic Boolean manipulation with ordered binary decision diagrams [J]. Journal of ACM Computing Surveys, 1992, 24(3):293-318

[14] Lin H, Sun J G. Knowledge Compilation Using the Extension Rule [J]. Journal of Auto Reasoning, 2004, 32(2):93-102

[15] Murray N V, Rosenthal E. Knowledge Compilation: Decomposable Negation Normal Form versus Linkless Formulas[C]// Proceedings of the Workshop on Model Computation-Principles, Algorithms, Applications, CADE-19. Miami, Florida, 2003, 18: 47-62

[16] Murray N V, Rosenthal E. Duality in Knowledge Compilation Techniques [J]. Journal of ISMIS, 2005:182-190

[17] Hähnle R, Murray N V, Rosenthal E. Normal Forms for Knowledge Compilation [J]. Journal of ISMIS, 2005:304-313

[18] Murray N V, Erik R. Updating Reduced Implicate Tries[C]// Proceedings of TABLEAUX 2007. 2007:183-198

[19] Darwiche A. On the tractability of counting theory models and its application to belief revision and truth maintenance [J]. Journal of Applied Non-Classical Logics, 2001, 11(1/2):11-34

[20] 李占山,姜云飞. 基于模型诊断推理的回顾与展望[J]. 计算机科学,1998,25(6):54-57

[21] Weld D. Recent Advances in AI Planning [J]. Artificial Intelligence Magazine, 1999, 20(2):93-123

[22] Huang J B. Combining Knowledge Compilation and Search for Conformant Probabilistic Planning [C] // Proceedings of ICAPS06. 2006:253-262

[23] Huang J B, Darwiche A. DPLL with a Trace: From SAT to Knowledge Compilation[C]// Proceedings of IJCAI 05. 2005: 156-162

[24] Geffner H. Planning Graphs and Knowledge Compilation[C]// Proceedings of ICAPS04. 2004:52-62

[25] Furbach U, Obermaier C. Knowledge compilation for description logics[C]// Proceedings of the 3rd Workshop on Knowledge Engineering and Software Engineering (KESK). 2007:418-420

[26] Meghyn B. Prime Implicate Normal Form for ALC Concepts[C]// Proceedings of AAAI 08. 2008:412-417

[27] Salem B, Henri P. Compiling Possibilistic Knowledge Bases[C]// Proceedings of ECAI06. 2006:337-341

[28] Cadoli M, Donini F M. A survey on knowledge compilation [J]. AI Communications, 1997, 10:137-150

[29] Boufkhad Y, Gregoire E, Marquis P, et al. Tractable cover compilations[C]// Proceedings of IJCAI97. 1997:122-127

[30] Kean A, Tsiknis G. Assumption based reasoning and clause management systems [J]. Computational Intelligence, 1992, 8 (1):1-24

[31] Przymusinski T C. An algorithm to compute circumscription [J]. Artificial Intelligence, 1989, 38:49-73

[32] Reiter R. A theory of diagnosis from first principles [J]. Artificial Intelligence, 1987, 32:57-95

[33] Coudert O, Madre J. Implicit and incremental computation of primes and essential implicant primes of boolean functions[C]// Proceedings of the 29th ACM/IEEE Design Automation Conference. 1992:36-39

[34] Kleer D J. An improved incremental algorithm for computing prime implicants[C]// Proceedings of the AAAI92. 1992: 780-785

[35] Jackson P, Pais J. Computing prime implicants[C]// Proceedings of the 10th International Conference on Automated Deductions. Kaiserslautern, Germany, July 1990. In Lecture Notes in Artificial Intelligence, Springer-Verlag, 1990, 449:543-557

[36] Kean A, Tsiknis G. An incremental method for generating prime implicants/implicates [J]. Journal of Symbolic Computation, 1990, 9:185-206

[37] Teow H N. A New Algorithm for Incremental Prime Implicate Generation[C]// Proceedings of IJCAI93. 1993:46-51

[38] Ramesh A, Murray N V. Parameterized Prime Implicant/Implicate Computations for Regular Logics [J]. Mathware & Soft Computing, Special Issue on Deduction in Many-Valued Logics, 1997, 2:155-179

[39] Slagle J R, Chang C L, Lee R C T. A new algorithm for generating prime implicants [J]. IEEE Transactions on Computers, 1970, 19(4):304-310

[40] Strzemecki T. Polynomial-time algorithm for generation of prime implicants [J]. Complexity, 1992, 8:37-63

[41] Raymond R, Johan D K. Foundations of Assumption-based Truth Maintenance Systems[R]. Preliminary Report of AAAI 87. 1987:183-189

[42] Murray N V, Rosenthal E. Efficient Query Processing with Compiled Knowledge Bases [C] // Proceedings of TABLEAUX 2005. 2005:231-244

[43] Morrison D R. PATRICIA — practical algorithm to retrieve information coded in alphanumeric [J]. ACM, 1968, 15(4):514-34

[44] Darwiche A. A Compiler for Deterministic, Decomposable Negation Normal Form[C]// Proceedings of AAAI/IAAI 02. 2002: 627-634

[45] Darwiche A. Compiling Knowledge into Decomposable Negation Normal Form[C]// Proceedings of IJCAI 99. 1999:284-289

[46] Dagostino M, Mondadori M. The taming of the cut [J]. Logic and Computation, 1995, 4(3):285-319

[47] Mondadori M. Classical Analytical Deduction, part I & II[Z]. Annali dell' University di Ferrara, Nuova Serie, sezione III, Filosofia, 1988:1-5

[48] Meghyn B. Prime Implicates and Prime Implicants in Modal Logic[C]// Proceedings of AAAI07. 2007:379-384

[49] Alvaro D V. Approximate Knowledge Compilation : The First Order Case[C]// Proceedings of AAAI96/IAAI. 1996:498-503

[50] Alvaro D V. An Analysis of Approximate Knowledge Compilation Proceedings of IJCAI95. 1995:830-836