ISSN 1000-9825, CODEN RUXUEW
*Journal of Software*, Vol.21, No.8, 2010, pp.1863−1877
doi: 10.3724/SP.J.1001.2010.03597
© by *Institute of Software*, *the Chinese Academy of Sciences*. All rights reserved.
E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

# 完全析取范式群判定 $\mathcal{SHOIN}(D)$-可满足性*

古华茂[1], 王 勋[1], 凌 云[1], 高 济[2+]

[1](浙江工商大学 计算机与信息工程学院,浙江 杭州 310018)

[2](浙江大学 人工智能研究所,浙江 杭州 310027)

## Determining the $\mathcal{SHOIN}(D)$-Satisfiability with a Complete Disjunctive Normal Form Group

GU Hua-Mao[1], WANG Xun[1], LING Yun[1], GAO Ji[2+]

[1](College of Computer and Information Engineering, Zhejiang Gongshang University, Hangzhou 310018, China)

[2](Institute of Artificial Intelligence, Zhejiang University, Hangzhou 310027, China)

+ Corresponding author: E-mail: gaoji@mail.hz.zj.cn

**Gu HM, Wang X, Ling Y, Gao J. Determining the $\mathcal{SHOIN}(D)$-satisfiability with a complete disjunctive normal form group. *Journal of Software*, 2010,21(8):1863−1877.** http://www.jos.org.cn/1000-9825/3597.htm

**Abstract**: This paper presents an approach to check the satisfiability of acyclic $\mathcal{SHOIN}(D)$-concepts—CDNF (complete disjunctive normal form) algorithm. This calculus can make a direct judgment on the satisfiability of acyclic $\mathcal{SHOIN}(D)$-concept by restructuring it into a hierarchical disjunctive normal form group on concept descriptions which is satisfiability self-telling, and reusing description clauses to block unnecessary extensions. CDNF algorithm eliminates description overlaps to the largest extent for it works on concept description directly. Therefore, CDNF algorithm has much better performance than Tableau as it saves a lot of spatial and temporal costs.

**Key words**: reasoning in description logic; satisfiability; disjunctive normal form; $\mathcal{SHOIN}(D)$; Tableau

**摘 要**: 针对非循环概念提出了一种对 $\mathcal{SHOIN}(D)$-概念可满足性进行判断的方法——CDNF(complete disjunctive normal form)算法.该算法通过把非循环定义的概念描述本身构建成分层次的析取范式群,并通过子句重用技术阻止无谓的子概念扩展,这样的析取范式群具有可满足性自明性,从而可以实现对 $\mathcal{SHOIN}(D)$-概念可满足性的直接判断.该算法基本上消除了判断过程中描述重复的现象,从而在空间、时间性能上都比 Tableau 算法有更好的表现.

**关键词**: 描述逻辑推理;可满足性;析取范式;$\mathcal{SHOIN}(D)$;Tableau

**中图法分类号**: TP182 **文献标识码**: A

在语义 WEB 体系结构中,本体(ontology)主要用于语义描述各种网络资源及资源间的关系与协作,是语义 WEB 研究的核心内容.无论是早期的语义 WEB 本体语言 OIL[1],DAML+OIL[2]还是现在的 OWL[3],OWL 2[4],都建立在描述逻辑基础之上.随着人们对信息智能化要求的不断提高,描述逻辑在其他智能应用中也将发挥越来

* Supported by the National Natural Science Foundation of China under Grant Nos.60775029, 60873218 (国家自然科学基金); the National High-Tech Research and Development Plan of China under Grant No.2007AA01Z187 (国家高技术研究发展计划(863)); the Zhejiang Provincal Natural Science Foundation of China under Grant No.Y1090734 (浙江省自然科学基金)

Received 2008-07-31; Revised 2008-11-27; Accepted 2009-02-16

© 中国科学院软件研究所 http://www.jos.org.cn

越重要的作用.而在描述逻辑的相关研究中,概念可满足性的推理是最基本的问题之一.

比较而言,基于归结的推理算法[5]虽然也能解决概念可满足性的判断问题,但其优势主要在于,在对含有大容量的 *ABox* 的情况下对实例检测及实例查询应答推理的效率要好于 Tableau 算法.单就一个概念的可满足性而言,其效率甚至还不如 Tableau,因为其还增加了一个归结过程.至于结构包含算法,因为它只适应于简单的描述逻辑语言(如 $FL_0$),基本上不具实用性.正因为如此,描述逻辑在概念可满足性推理方面的理论研究和系统开发应用基本上都是围绕着经典概念可满足性算法 Tableau 展开的.要么在已有描述逻辑系统中增加新的算子或功能,同时对 Tableau 算法进行一定的扩展以适用新的系统[6−8];要么把描述逻辑与其他研究领域结合起来从而赋予描述逻辑新的研究内容,但是在概念可满足性推理方面依然是在 Tableau 上进行扩展[9−12].

事实上,虽然 Tableau 推理算法经实践证明是可行的,但由于其实现机理等因素而存在一些自身无法逾越的缺陷.由于 Tableau 算法是通过对一个概念 $C_0$ 的 $ABox\{C_0(x)\}$,根据 $x$ 必须满足的条件,按照相应的规则进行层层展开,最终以一致或不一致的 *ABox* 来判断概念 $C_0$ 的可满足性.而每一次展开实际上都是对被展开概念的全部或部分重复.例如,如果有$(A\cap B)(x')\in ABox$,则根据 Tableau 算法的$\cap^-$规则,必有:$(A)(x')\in ABox$ 且$(B)(x')\in ABox$,这是完全重复的情况.同理,如果有$(A\cup B)(x')\in ABox$,则必有$(A)(x')\in ABox$ 或$(B)(x')\in ABox$,这是部分重复的情况.再如,如果有$(\exists R.C)(x')\in ABox$,则根据$\exists^-$规则,必有:$(C)(x')\in ABox$,这也是近乎完全重复的情况.而$\cap$,$\cup$,$\exists$是概念表示中出现频率相当高的算子,所以这种重复现象也就非常普遍.

本文提出的 CDNF(complete disjunctive normal form)算法就是针对 Tableau 算法中概念描述重复严重,浪费了大量空间这种现象而提出的一种算法.其基本思想是,通过对非循环定义的概念描述[13]本身在不同层次上构建析取范式的方式进行重新组合,重组过程中不改变概念的模型特性,即在每一步重组中,后生成的概念如果可满足,则重组前的概念也是可满足的.然后根据所获得的新的概念描述直接判断原概念描述的可满足性.CDNF 算法在对概念的处理不存在子概念的重复,从而相对于 Tableau 而言节省了大量的空间,使得 CDNF 算法在大部分情况下可以取得线性空间的效果.对于含有循环定义的概念,目前的做法是,要么采用一般包含公理形式来表示,即不再作为一个独立的概念定义的形式出现,要么通过与命题模态逻辑的联系,利用其命题 $\mu$-calculus 实现推理,而这二者都不在本文的讨论范围.除非特别说明,本文后面提到的概念均是非循环的.

OWL DL[14]是目前应用最广泛的 WEB 本体表示语言.它的逻辑基础是描述逻辑语言 $\mathcal{SHOIN}(D)$[14].因此,我们选用 $\mathcal{SHOIN}(D)$作为目前 CDNF 算法的实现对象.本文首先介绍 $\mathcal{SHOIN}(D)$的语法语义,然后详细介绍其对应的 CDNF 算法过程、相关证明以及性能分析,最后对 CDNF 与 Tablueau 算法的性能作一个简单的比较.

## 1 $\mathcal{SHOIN}(D)$的语法与语义

描述逻辑是通过一组构建算子对领域知识进行建模,以概念和角色的形式表示知识并在此基础上实现推理的形式化方法.描述逻辑 $\mathcal{SHOIN}(D)$是在描述逻辑语言 $\mathcal{SHIN}$ 的基础上充分考虑本体表示的需要,从而支持具体数据类型[15](concrete datatypes)以及命名个体(named individual)[16],具有很强的表达能力的面向本体的描述逻辑语言.具体数据类型使得概念描述中可包含如数值,字符串等数据类型.命名个体则可以用于表示具体的人、公司、国家,并出现在概念描述中.这两者都会在本体中经常用到.下面对 $\mathcal{SHOIN}(D)$语法和语义作一个简单的介绍.

**定义 1**($\mathcal{SHOIN}(D)$-概念). 设 $D$ 是数据类型集合,$NC$ 是 $\mathcal{SHOIN}(D)$-概念名集合,$NI$ 是个体名集合(不同个体名有可能代表同一个体),$NR=N_R^A \uplus \{R^-|R\in N_R^A\} \uplus N_R^D$ 是 $\mathcal{SHOIN}(D)$角色集合($N_R^A$是抽象角色名集,$N_R^D$是具体角色名集,$NR^+\subseteq N_R^A$是传递角色名集合),那么,$\mathcal{SHOIN}(D)$概念集合是满足下列条件的最小集合:

1. 任何概念名 $C\in NC$ 是 $\mathcal{SHOIN}(D)$-概念;
2. 对任何个体名 $o\in NI$,$\{o\}$是 $\mathcal{SHOIN}(D)$-概念;
3. 若 $C$,$D$ 是 $\mathcal{SHOIN}(D)$-概念,$R\in N_R^A \uplus \{R^-|R\in N_R^A\}$是抽象角色,$F\in N_R^D$是具体角色,$S\in N_R^A \uplus \{R^-|R\in N_R^A\}$是简单角色(抽象角色中,自身没有传递性且不含传递性的子角色)[17],$d\in D$ 是数据类型,$n\in IN$(自然数),则$(C\cup D)$,$(C\cap D)$,$(\neg C)$,$d$,$(\neg d)$,$(\forall R.C)$,$(\exists R.C)$,$(\geqslant nS)$,$(\leqslant nS)$,$(\forall F.d)$,$(\exists F.d)$都是 $\mathcal{SHOIN}(D)$-概念.

© 中国科学院软件研究所 http://www.jos.org.cn

此外,⊤称为全概念,代表包含所有概念的合集;⊥称为空概念,代表不包含任何个体,是任何概念的子概念.我们把数量限制限定于简单角色主要是为了确保可判断性[18].

**定义 2(*TBox*)**. 假设 $C,D$ 是概念,则形如 $C\subseteq D$ 的概念表达式称为概念包含公理;形如 $C\equiv D$ 的概念表达式称为概念等价公理,更进一步地,如果 $C$ 为概念名,则它又称为概念定义公理,此时 $D$ 往往称作 $C$ 的概念描述.一个描述逻辑术语库 *TBox* $\mathcal{T}$ 一般为一组概念包含公理及概念等价公理的集合.如果 $A\in NC$,并且 $A$ 只出现在概念定义的右侧,即 $A$ 不能再由其他概念通过算子组合而成,则 $A$ 称为原始概念[13].

因为一条概念等价公理可转换成两条概念包含公理,而概念包含公理在概念可满足性推理中需要特殊的处理,所以为了便于描述,我们先假设 *TBox* 中只含有概念定义公理,而且这些定义都是非循环的[13].此外,值得注意的是,在本文中经常可见到形如"$C$ 是概念"或"$\exists R.C$"等表述,这里的 $C$ 显然未必是概念名(于是未必在 *TBox* 中存在一条对应的定义公理),此时,$C$ 代表了一段概念描述,因此 $C$ 可视作是一个临时的概念名,而那段概念描述正好就是对 $C$ 的定义.

**定义 3(*RBox*)**. 假设 $R,S$ 是两个抽象或具体角色,则形如 $R\subseteq S$ 的角色表达式称为角色包含公理.一个描述逻辑角色公理集 *RBox* 一般为一组角色包含公理的集合.对于某个 *RBox*,我们称 $RBox^+$ 为它的角色分层:$RBox^+:=(RBox\cup\{Inv(R)\subseteq Inv(S)|R\subseteq S\in RBox\},\sqsubseteq)$,$\sqsubseteq$ 是 $\subseteq$ 在 $RBox\cup\{Inv(R)\subseteq Inv(S)|R\subseteq S\in RBox\}$ 上的自反传递闭包.

为了便于描述,我们引入两个角色函数:

(1) 因为角色的逆运算具有对称性,为了避免出现形如 $R^{-}$ 的角色表示,我们定义函数 *Inv* 用于返回角色的逆:如果 $R$ 是角色名,那么 $Inv(R)=R^-$;如果 $R$ 已经是一个角色名 $S$ 的逆形式,则 $Inv(R)=S$.

(2) 如果一个角色 $R$ 是传递的,那么 $Inv(R)$ 也是传递的,反之亦然.但出现在传递性角色名集合 $NR^+$ 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,即 $Trans(R)=true$ 当且仅当 $R\in NR^+$ 或 $Inv(R)\in NR^+$.

**定义 4($\mathcal{SHOIN}(D)$-解释)**. $\mathcal{SHOIN}(D)$-解释 $I=(\Delta^I,\cdot^I)$ 由非空集合 $\Delta^I$(解释域)和解释函数 $\cdot^I$ 组成.$\Delta^D$ 是所有数据类型的论域,并且论域 $\Delta^D$ 与 $\Delta^I$ 不相交.对于每个数据类型 $d$,其数值范围 $d^D\subseteq\Delta^D$,并且有 $(\neg d)^D=\Delta^D\backslash d^D$.函数 $\cdot^I$ 将每个概念 $A$ 映射为子集 $A^I\subseteq\Delta^I$,将每个抽象角色 $R$ 映射为子集 $R^I\subseteq\Delta^I\times\Delta^I$,将每个具体角色 $F$ 映射为子集 $F^I\subseteq\Delta^I\times\Delta^D$,将每个命名个体的概念 $\{o\}$ 映射为 $\{o\}^I\subseteq\Delta^I$ 且 $\#\{o\}^I=1$(#代表一个集合的势).此外,解释函数 $\cdot^I$ 还应该满足:

- $(C\cap D)^I=C^I\cap D^I;(C\cup D)^I=C^I\cup D^I;(\neg C)^I=\Delta^I\backslash C^I$.
- $(\exists R.C)^I=\{x\in\Delta^I|$ 存在一个 $y\in\Delta^I$,满足 $\langle x,y\rangle\in R^I$,且 $y\in C^I\}$;$(\forall R.C)^I=\{x\in\Delta^I|$ 对任意 $y\in\Delta^I$,如果 $\langle x,y\rangle\in R^I$,那么 $y\in C^I\}$.
- $(\geqslant nS)^I=\{x\in\Delta^I|\#\{\langle x,y\rangle|\langle x,y\rangle\in S^I\}\geqslant n\}$;$(\leqslant nS)^I=\{x\in\Delta^I|\#\{\langle x,y\rangle|\langle x,y\rangle\in S^I\}\leqslant n\}$.
- $(\exists F.d)^I=\{x\in\Delta^I|$存在一个 $y\in\Delta^D$,满足 $\langle x,y\rangle\in F^I$,而且 $y\in d^D\}$;$(\forall F.d)^I=\{x\in\Delta^I|$对任意 $y\in\Delta^D$,如果 $\langle x,y\rangle\in F^I$,那么 $y\in d^D\}$.
- 对任意 $S\in N_R^A\cup\{R^-|R\in N_R^A\}$,$\langle x,y\rangle\in S^I$,当且仅当 $\langle y,x\rangle\in S^{-I}$.
- 对任意 $R\in NR^+\cup\{P^-|P\in NR^+\}$,如果有 $\langle x,y\rangle\in R^I$,$\langle y,z\rangle\in R^I$,那么一定有 $\langle x,z\rangle\in R^I$.

假设 $I$ 是一个解释,如果有 $C^I\neq\varnothing$,则称 $I$ 满足概念 $C$ 并且 $C$ 是可满足的;如果有 $C^I\subseteq D^I$(或 $C^I\equiv D^I$),则称 $I$ 满足概念公理 $C\subseteq D$(或 $C\equiv D$);如果有 $R^I\subseteq S^I$,则称 $I$ 满足角色公理 $R\subseteq S$.如果 $I$ 满足 *RBox* 中的所有角色公理,则 $I$ 称为该 *RBox* 的模型.如果 $I$ 满足 *TBox* 中的所有概念公理,则 $I$ 称为该 *TBox* 的模型.如果存在 *TBox*(与/或 *RBox*)的模型 $I$ 使得 $C^I\neq\varnothing$,则称概念 $C$ 是关于 *TBox*(与/或 *RBox*)可满足的,$I$ 也称为 $C$ 关于 *TBox*(与/或 *RBox*)的模型.

因为这里的概念 $C$ 为非循环概念,所以可把 $C$ 的定义中出现的概念名替换成它们在 *TBox* 中对应的概念定义部分,直至 $C$ 的定义中不再出现非原始概念,这样,概念 $C$ 关于 *TBox* 和 *RBox* 的可满足性即可转化为关于空 *TBox* 和 *RBox* 的可满足性,即仅需判定 $C$ 是否关于 *RBox* 可满足即可.更详细的描述参见描述逻辑手册[13].

$\mathcal{SHOIN}(D)$ 的否定算子 $\neg$ 使得包含关系的推理和概念可满足性的判定之间可以互相转化.这是因为:$C\subseteq D$ 成立当且仅当 $C\cap\neg D$ 是不可满足的;反之,$C$ 是可满足的当且仅当 $C\subseteq\bot$ 不成立.因此,下面的讨论中只涉及概念的可满足性判定问题.

© 中国科学院软件研究所 http://www.jos.org.cn

## 2 面向 $\mathcal{SHOIN}(D)$的 CDNF 算法

CDNF 算法采用完全不同于 Tableau 的算法思路,直接在概念描述上构建完全析取范式来判断概念的可满足性.本节先给出相关的基本定义,然后从 CDNF 构建过程、算法可靠性、完备性与终止性等方面全面论述 CDNF 算法.

### 2.1 相关基本定义

**定义 5(文字和限定概念)**. 假设 $A$ 是原始概念,$o$ 为个体名,$C$ 是概念,$R$ 为抽象角色,$S$ 为简单角色,$F$ 为具体角色,$n$ 是正整数,则 $A,\neg A,\{o\},\exists R.C,\forall R.C,(\geqslant nS),(\leqslant nS),(\forall F.d),(\exists F.d),d,\neg d,\top,\bot$称为文字.另外,$\exists R.C,\forall R.C$称为∃/∀角色文字(统称为角色文字),或更具体地称为∃/∀$R$角色文字,$C$分别称为角色文字$\exists R.C$和$\forall R.C$的限定概念.在本文中,文字我们一般用大写的 $X,Y$ 表示.例如,对概念描述$\exists R.(A\cap(\exists F.d))\cup\exists S.C\cap A$而言,它由 3 个文字组成,即$\exists R.(A\cap(\exists F.d)),\exists S.C$和$A$.而对其第 1 个文字的限定概念而言,它由两个文字组成,即 $A$ 和$(\exists F.d)$.因此,这里任何概念描述都可以解析成有限个文字.

**定义 6(子句)**. 子句是满足下面条件的概念描述:单个文字是子句;两个或多个文字通过∩算子连在一起形成的概念描述也是子句,在本文中,子句一般用小写的字母表示,如 $x,y,z$ 等.∃角色文字的限定概念经过 CDNF 算法重组后会形成一个子句,所以限定概念在形成子句后,又称为限定概念子句.如果子句中含有角色文字,则角色文字的限定概念也称为子句的限定概念.反过来,子句称为这些限定概念(子句)的父子句,角色文字的角色称为其限定概念的父角色.父子句与子句之间可通过 $L(x,y)=R$ 表示,即 $x$ 是 $y$ 的父子句.$x$ 可通过 $y$ 表示为 $\mathcal{U}pper(y)$,而 $y$ 是 $x$ 中的某个∃$R$ 文字的限定概念子句,即 $y$ 是 $x$ 的子子句.产生子句 $y$ 的∃角色文字可表示为 $y.Letter$,显然,$y.Letter$ 是子句 $x$ 的文字之一.一个文字 $X$ 在子句 $x$ 中出现,我们表示为 $X\in x$.例如,概念描述$\exists R.(A\cap(\exists F.d))\cap\exists S.C$,这显然是一个子句(设标记为 $x$),而其第 1 个文字的限定概念也是一个子句(设标记为 $y$).由此可得:$\mathcal{L}(x,y)=R,\mathcal{U}pper(y)=x=\exists R.(A\cap(\exists F.d))\cap\exists S.C,y.Letter=\exists R.(A\cap(\exists F.d))$,文字$\exists S.C\in x$等.

**定义 7(析取范式)**. 由单个子句组成或由两个及两个以上子句通过∪算子连接在一起的概念描述称为析取范式(DNF).例如,概念描述$\exists R.(A\cap(\exists F.d))\cup\exists S.C\cup A$就是一个由 3 个子句构成的析取范式.

**定义 8(完全析取范式)**. 如果一个概念描述是一个析取范式(我们称其为顶层析取范式),而且组成这个析取范式的所有∃角色文字的限定概念也都是完全析取范式,则该概念称为完全析取范式(CDNF).注意,CDNF 在本文中根据上下文即表示算法名,又代表实际的一个完全析取范式.例如,概念描述$\exists R.(A\cap(\exists F.d))\cup\exists S.A$满足完全析取范式的格式,而概念描述$\exists R.(A\cap(B\cup C))\cup\exists S.A$则不满足,因为其第 1 个文字的限定概念不是析取范式.

**定义 9(概念描述群与 CDNF 群)**. 对于一个含命名个体的概念描述,如果它是可满足的,则往往要求这些命名个体属于某些概念的实例.这样,每个命名个体都有一个对应的概念描述(对命名个体 $o$,其表示为 $Des(o)$),于是与前面的初始概念描述一起就形成了概念描述群.这个初始的概念描述称为主概念描述.这个概念描述群经 CDNF 算法后自然会形成一组完全析取范式,称为 CDNF 群.同样,主概念描述对应的完全析取范式称为主 CDNF.

**定义 10(子句标记集、初始限定概念标记、∃角色标记集)**. 每个子句 $x$ 都有一个标记集 $\mathcal{B}$,用于存放从子子句反作用过来的概念,表示为 $\mathcal{B}(x)$;如果一个∃角色文字的限定概念中含有传递性角色(或传递性角色的子角色),则为其提供一个标记数据结构 $\mathcal{E}$ 用于存放它的初始限定概念,用于后面及时终止不必要的扩展.文字 $X$ 的初始限定概念表示为 $\mathcal{E}(X)$;每个简单角色的∃角色文字有一个标记集 $\mathcal{R}$,用于存放自己必须支持的角色集,表示为 $\mathcal{R}(X)$.如文字 $X=\exists R.B$,而 $\mathcal{R}(X)=\{R,S,Inv(P)\}$,则意味着文字 $X$ 不仅代表$\exists R.B$,还同时代表着$\exists S.B$ 和$\exists Inv(P).B$,所以,如果 $X\in$子句 $x$,子句 $y$ 是概念描述 $B$ 经算法处理所得的子句,则 $\mathcal{L}(x,y)=R,\mathcal{L}(x,y)=S,\mathcal{L}(x,y)=Inv(P)$同时成立,此时,$\mathcal{L}(x,y)$不再对应于一个角色,而是一组角色,写作 $\mathcal{L}(x,y)=\mathcal{R}(X)=\{R,S,Inv(P)\}$.为了统一表示,可把 $\mathcal{R}(X)$显式地嵌在 $X$ 中,如前面的例子,直接表示为$\exists(R,S,Inv(P)).B$ 即可.此外,本文假设非简单角色的∃角色文字同样有一个标记集 $\mathcal{R}$,只不过该标记集只含有 1 个元素,即文字上所用角色本身.

**定义 11(后继、前趋、邻居)**. 假设角色 $R,S$ 之间满足 $R\sqsubseteq S$.对于一个子句 $y$,如果∃角色文字 $X\in y$,并且 $R\in\mathcal{R}(X)$,

© 中国科学院软件研究所 http://www.jos.org.cn

则 $X$ 的限定概念称为 $y$ 的 $S$-后继.对子句 $x$ 和 $y$,如果 $R\in\mathcal{L}(x,y)$,则子句 $x$ 称为子句 $y$ 的 $Inv(S)$-前趋(显然,$y$ 是子句 $x$ 的 $S$-后继).子句 $y$ 的 $S$-后继或 $S$-前趋统称为 $y$ 的 $S$-邻居.

**定义 12(逆 Merge 操作)**. 假设子句 $x$ 是子句 $y$ 的 $R$-前趋,角色 $R,S$ 之间满足 $R\sqsubseteq S$,则:对于任意文字 $\forall S.C\in y$,根据语义,必须将 $\forall S$ 文字作用于 $x$ 子句中:如果 $Trans(R)$ 为假,则原 $x$ 子句与 $\forall S.C$ 文字的限定概念 $C$ 通过 $\cap$ 算子连在一起形成新的概念描述,即 $x\cap C$,取代原来的 $x$;否则,即 $Trans(R)$ 为真,将原 $x$ 子句与 $\forall R.C$ 文字及其限定概念 $C$ 通过 $\cap$ 算子连在一起形成新的概念描述,取代原来的 $x$,即 $x\cap C\cap\forall R.C$.然后,无论是否 $Trans(R)$ 为真,都把 $\forall S.C$ 文字添加 $\mathcal{B}(x)$ 中(即让 $\forall S.C\in\mathcal{B}(x)$).这个操作称为逆 $\forall$-Merge 操作.

另外,如果 $y$ 子句中含有 $\leqslant mS$ 型文字($S$ 在此为简单角色),而 $y$ 中含有的 $S$-邻居数(如 $n$)大于 $m$,则要对 $y$ 的一些 $S$-邻居进行合并,以达到 $\leqslant mS$ 型文字的要求.如果合并是把一个邻居合并到父子句 $x$ 中,即 $x$ 子句与 $y$ 子句中某个 $X=\exists P.D$ 文字(角色 $P,S$ 之间满足 $P\sqsubseteq S$)的限定概念 $D$ 通过 $\cap$ 算子连在一起形成新的概念描述:$x\cap D$,同时把该 $\exists P$ 文字的限定概念 $D$ 添加到 $\mathcal{B}(x)$ 中,并从 $y$ 子句中移去文字 $\exists P.D$,让 $\mathcal{R}(y.Letter)=\mathcal{R}(y.Letter)\cup\{Inv(Q)|Q\in\mathcal{R}(X)\}$,我们也称其为逆 $\exists$-Merge 操作.

**定义 13(路径)**. 在 CDNF 中,子句序列 $x_1,x_2,\ldots,x_n,x_i$ 是 $x_{i+1}$ 的父子句($i\geqslant 1$ 并且 $i\leqslant n-1$),我们称这样的序列为一个路径.而且对于某个 $x_j(1<j\leqslant n)$,所有 $x_i(0<i<j)$ 都称为 $x_j$ 的祖先子句,对于某个 $x_i(1\leqslant i<n)$,所有 $x_j(i<j\leqslant n)$ 都称为 $x_i$ 的子孙子句.

**定义 14(子句重用)**. 假设存在一条路径 $x_1,x_2,\ldots,x_n$,如果存在两子句 $x_i,x_j(j>i)$ 满足:

$\mathcal{R}(x_i.Letter)=\mathcal{R}(x_j.Letter),\mathcal{E}(x_i.Letter)=\mathcal{E}(x_j.Letter)$,并且 $\mathcal{U}pper(x_i)-x_i.Letter=\mathcal{U}pper(x_j)-x_j.Letter,x_i/\mathcal{R}=x_j/\mathcal{R}$,

其中,集合 $\mathcal{U}pper(x)-x.Letter$ 定义为 $\mathcal{U}pper(x)-x.Letter=\{X|X\in\mathcal{U}pper(x)$,但 $X\neq x.Letter\}\}$,$x/\mathcal{R}$ 定义为 $x/\mathcal{R}=\{\forall Inv(S).C|\forall Inv(S).C\in x$,且 $R\sqsubseteq S,R\in\mathcal{R}(x.Letter)\}$,那么,$x_i$ 为可被 $x_j$ 重用的子句($x_i$ 称为重用子句,而 $x_j$ 称为备用子句).此时,从 $x_j.Letter$ 文字连一根有向重用线到 $x_i.Letter$ 的限定概念子句(即 $x_i$)上,表示 $x_i$ 就是角色文字 $x_j.Letter$ 的限定概念子句.而 $x_j$ 以及 $x_j$ 的子孙子句都不再进行展开.但是要注意的是,$x_j/\mathcal{R}$ 的文字要作用于子句 $\mathcal{U}pper(x_j)$ 上.而且一旦上面的条件不再满足(由于其他子句的作用所致),则这根重用线自动消失,而 $x_j$ 又需要继续展开了,这种情况可称为动态重用.

**定义 15(Merge 操作)**. Merge 操作是指将两个角色文字合并为一个角色文字的操作.假设 $B,C$ 是概念,$R$ 是角色(包括抽象角色和具体角色),则我们进行如下操作:

对于子句 $y$ 的 $R$-后继 $B$,如果有文字 $\forall S.C\in y$ 并且 $R\sqsubseteq S$,那么,如果 $Trans(R)$ 为假,则以 $B\cap C$ 替换 $B$;如果 $Trans(R)$ 为真,则以 $(B\cap C\cap\forall R.C)$ 替换 $B$.这称为 $\forall$-Merge 操作.一个子句中的所有 $\forall$ 角色文字都要通过 $\forall$-Merge 操作结合到所有关于该角色的后继中去.

假设文字 $X=\exists R.B,Y=\exists S.C$,则将 $\exists R.B\cap\exists S.C$ 合并为文字 $Z=\exists R.(B\cap C)$ 替换原来的文字 $\exists R.B$ 和 $\exists S.C$,并且让 $\mathcal{R}(Z)=\mathcal{R}(X)\cup\mathcal{R}(Y)$,这称为 $\exists$-Merge 操作.这个操作主要是通过合并一些 $\exists$ 角色文字以便在生成解释时可满足最大数量限制文字的语义要求.

如果一个子句 $x$ 中出现形如 $B\cap\{o_1\}$ 的概念描述,则将概念 $B$ 转移到命名个体 $o_1$ 对应的概念描述,形成 $Des(o_1)=Des(o_1)\cap B$,同时留下 $\{o_1\}$ 在原来位置,即 $x=\{o_1\}$.如果 $x$ 是另外某命名个体(如 $o_2$)对应的概念描述(即 $x=Des(o_2)$),则标记 $o_1=o_2$ 表示这两个个体实际为同一个体.这称为 $o$-Merge 操作.

**定义 16(处理点)**. 在把一个概念描述转化为完全析取范式的过程中,不仅需要将该概念转化为析取范式,而且要将其中的限定概念也转化为析取范式,直到各层的限定概念都转化为析取范式为止.因此,我们把在 CDNF 算法执行过程中即将被转化为析取范式或当前正在形成析取范式的子概念对象称为处理点.处理点中当前执行算法操作的子句称为处理子句.

### 2.2 CDNF算法过程

以概念 $C_0$ 的描述作为输入,经过 CDNF 算法过程,可以直接获得 $C_0$ 可满足性的判断.如果 $C_0$ 可以获得至少 1 个无语义冲突的 CDNF 群,则 $C_0$ 是可满足的,否则就是不可满足的.下面将描述 CDNF 算法的过程(如图 1 所示,图中 $P$ 表示处理点).

© 中国科学院软件研究所 http://www.jos.org.cn

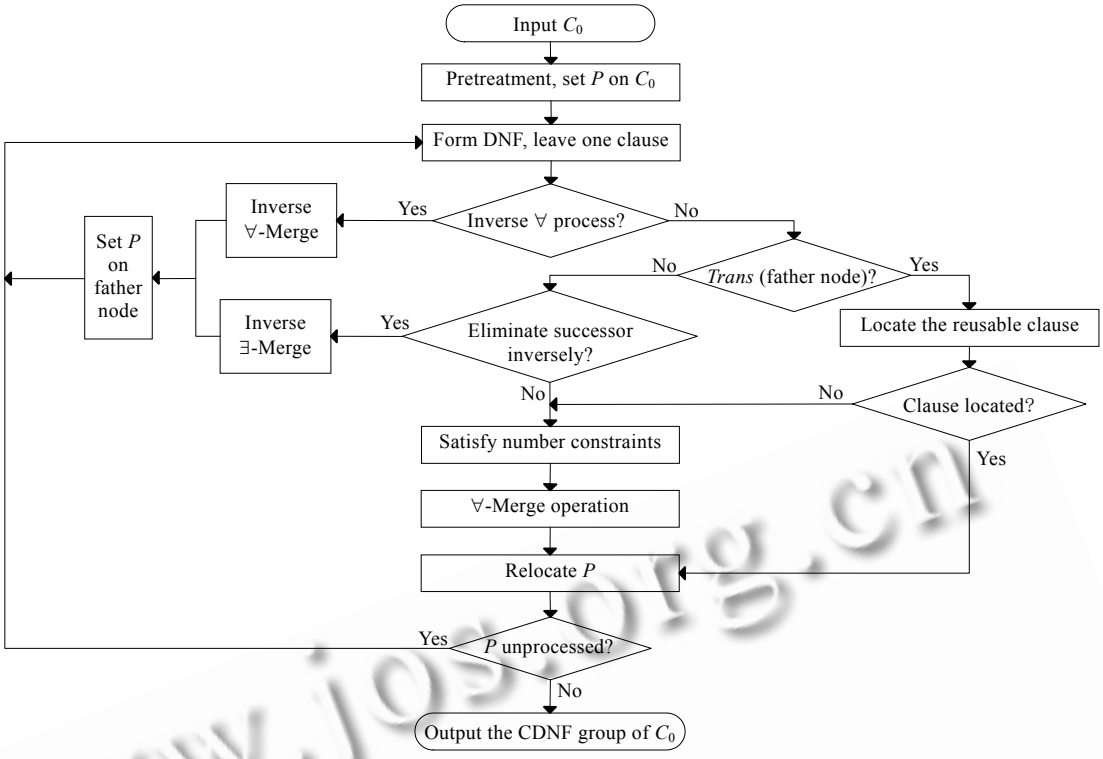


Fig.1 Process of CDNF algorithm to build a CDNF group

图 1 CDNF 算法形成 CDNF 群的过程

(1) 预处理.将 $C_0$ 中出现的所有概念名,除原始概念外都替换成它们对应的概念描述,直至 $C_0$ 中出现的概念名都属于原始概念.然后把 $C_0$ 转化为否定范式,即把所有的否定算子尽可能地往里推,使得否定算子只出现在原始概念名之前.这通过德·摩根律,以及 $\neg\exists R.C=\forall R.(\neg C)$, $\neg\forall R.C=\exists R.(\neg C)$等算子运算规则是容易实现的.数量限制文字的否可以通过更改数字与变更限制方向很容易地实现,如 $\neg(\leqslant nR)=\geqslant(n+1)R$,反之亦然.然后将 $C_0$ 设为处理点 $P$.同时,针对每个个体 $o$,初始化 $Des(o)=\top$,这些 $Des(o)$同样称为孤点(distinguished point)[17].此外,初始化两个二元关系(=和≠,分别存放等价命名个体对和不可合并∃角色文字对)为空集,对于概念描述中的每个∃角色文字$\exists R.C$,初始化其 $R$ 标记集仅包含 $R$.

(2) 形成析取范式.将处理点的文字通过∩,∪算子的结合律、交换律、分配律、幂等律等运算法则转化为析取范式.如果同一子句中存在两个以上相同的文字并且它们之间不存在≠关系,则保留一个即可.选取其中一个子句,其他子句则全部删除.检查选取的子句是否存在如下 3 类文字冲突:

① 概念名冲突: $\neg A\cap A$,$A$ 是原始概念.注意空概念⊥等价于 $\neg A\cap A$.

② 数据类型冲突:子句包含数据类型 $d_1,\ldots,d_n$,但 $d_1^D\cap\ldots\cap d_n^D=\varnothing$.

③ 命名个体冲突:对某个命名个体 $o$,存在$\neg\{o\}\in Des(o)$(此时 $Des(o)$应满足子句形式).

如果子句中不存在冲突,就检查本子句中是否含有$\{o\}$型文字,如果含有,则执行 $o$-Merge 操作.只要子句出现冲突,就把它替换为空概念⊥,标记该 CDNF 为⊥(即不可满足),所以该 CDNF 群也是不可满足的,因为根据规则: $\exists R.\bot\equiv\bot$及 $C\cap\bot\equiv\bot$($C$ 为概念描述),父子句也将被替换为⊥,直至最顶层的子句.这并不表示算法结束,因为可能还有其他 CDNF 群是可满足的.

注意,在定义 5 中已经罗列构成一个 $\mathcal{SHOIN}(D)$-概念的所有可能的文字类型.不考虑角色文字的限定概念,则构建一个复杂概念的唯一途径就是通过∩,∪算子以及辅助符号(如圆括号)把多个文字连结在一起.而根据语

© 中国科学院软件研究所 http://www.jos.org.cn

义,∩,∪算子支持形成析取范式所需的结合律、交换律、分配律、幂等律等.这样,对于一个概念描述,总是可以将其组织成一个析取范式.

(3) 逆∀处置.假设当前子句为 $x,y$ 为其 $R$-前趋,如果有文字$\forall S.C\in x$(角色 $R,S$ 之间满足 $R\sqsubseteq S$),并且$\forall S.C\notin\mathcal{B}(y)$,则要执行逆∀-Merge 操作,即根据 $R$ 的传递性,执行子概念重组:$C\cap y$ 或 $y\cap C\cap\forall R.C$.同时把$\forall S.C$ 添加到 $\mathcal{B}(y)$中.并把 $y$ 设为处理点 $P$,返回到步骤(2).在进行此项操作时,注意 $\mathcal{R}(x.Letter)$中的传递性角色要先处理.

(4) 逆消后继.假设当前子句为 $x,y$ 为其 $R$-前趋,而 $x$ 含数量约束$\leqslant nS$,并且角色 $R,S$ 之间满足 $R\sqsubseteq S$,设 $m=|\{\exists P.C|\exists P.C\in x,P\sqsubseteq S,C\notin\mathcal{B}(y)$或 $Inv(P)\notin\mathcal{R}(x.Letter)\}|+1$.因为在后面生成解释时,$x$ 与它的父子句会形成一对$(x,y)\in S^I$,所以父子句 $y$ 也应该看作是 $x$ 的一个 $S$ 的∃角色文字的限定概念.如果 $m>n$,则可以采用两种方式进行处理,一是从集合$\{\exists P.C|\exists P.C\in x,P\sqsubseteq S,C\notin\mathcal{B}(y)$或 $Inv(P)\notin\mathcal{R}(x.Letter)\}$中选取一个文字$\exists P.C$ 通过逆∃-Merge 操作合并到 $y$ 中,并把 $y$ 置为当前处理点,返回到步骤(2),这相当于消除了一个 $S$-后继.二是通过后面的≤满足步骤来达到 $m\leqslant n$ 的要求.

(5) 重用子句定位.如果当前子句 $x$ 的父角色 $R$ 为传递性角色或为某传递性角色的子角色,则从当前子句开始,沿着路径向上搜索可重用子句.如果定位成功,则不再需要对当前子句进行处理,把 $\mathcal{U}pper(x)$置为 $P$,并直接转到 $P$ 重定位步骤.

(6) 数量限制满足.假设当前子句为 $x$,并且有文字$\geqslant nS\in x$ 或$\leqslant nS\in x,y$ 为其 $R$-前趋.如果 $R\sqsubseteq S$,则从子句 $x$ 中清除所有文字$\{\exists P.C|\exists P.C\in x,P\sqsubseteq S,C\in\mathcal{B}(y)$且 $Inv(P)\in\mathcal{R}(x.Letter)\}$(因为这些都是可消除的 $S$-后继),然后令 $m=|\{\exists P.C\,|\,\exists P.C\in x,P\sqsubseteq S\}|+1$,否则令 $m=|\{\exists P.C|\exists P.C\in x,P\sqsubseteq S\}|$.如果 $x$ 中同时存在角色 $S'$的数量限制文字,且有 $S'\sqsubseteq S$,则一定要先处理 $S'$的数量限制文字:

① 如果是文字$\geqslant nS\in x$,且 $m<n$,则往 $x$ 里添加 $n$ 个∃$S$ 角色文字:$\exists S.\top$,同时标记这些新增文字之间相互不可合并(即≠).这个过程也称为≥满足操作.

② 如果文字$\leqslant nS\in x$,且 $m>n$,则在 $x$ 中任选不存在≠关系的两个 $S$-后继进行∃-Merge 操作,直至 $m=n$,或者再无可合并的 $S$-后继,但仍然有 $m>n$(这种情况代表冲突发生,置该 CDNF 为⊥).完成所有其他数量限制文字.这个过程也称为≤满足操作.

(7) ∀-处置.设当前子句为 $x$,进行如下处理:对任意文字$\forall S.C\in x$,把$\forall S.C$ 通过∀-Merge 操作结合到每个 $R$-后继 $B$ 中(其中 $R\sqsubseteq S$):根据 $R$ 的传递性,执行限定概念重组:$(B\cap C)$或$(B\cap C\cap\forall R.C)$.

(8) 处理点重定位.如果处理点不存在未处理的限定概念,则沿着父子句一直往上回溯.在回溯过程中,一定要检查经过的每个子句是否存在冲突:子句 $x$ 中存在两个∃角色文字,它们的限定概念均含有同一命名个体$\{o\}$或分别含有一个命名个体$\{o_1\}$和$\{o_2\}$并且有 $o_1=o_2$,但这两个文字存在≠关系.有则置当前 CDNF 为⊥.直至碰到含有未处理限定概念的祖先子句或最顶层的子句,把它设为处理点 $P$.如果回溯之后处理点中仍然没有未处理的限定概念,则转到下一个没有形成 CDNF 的孤点进行展开(注意,一个已经形成 CDNF 的孤点有可能因 $o$-Merge 操作而添加新的描述从而需要进一步处理,当然前面对该孤点所作的操作仍然有效),直至整个 CDNF 群生成完毕,算法结束.否则,从处理点中选取一个限定概念作为新的处理点,重复步骤(2)~步骤(7).这里要注意的是,如果当前子句的某∃文字的限定概念子句已经处理过,而当前子句的变化会影响到该限定概念子句,如产生了新的同角色∀文字,则它要作为未处理限定概念对待.此外,如果当前子句为某个子孙子句所重用,而当前子句的变化破坏重用条件,则重用应该撤销,该子孙子句也应该继续处理.但是,即便出现这两种情况,前面对该子孙子句所作的操作依然有效.

如果概念 $C_0$ 能够获得一个顶层析取范式均不为⊥的 CDNF 群,则 $C_0$ 是可满足的,否则就是不可满足的.

假设目前有个在线软件系统用户管理有以下要求:(1) 用户种类按级别从低到高:一般用户、管理员、超级管理员;(2) 低级用户不可以给高级用户授权,同级则可以,超级管理员可以给自己授权;(3) 授权是一种管理行为,任何管理行为的管理者必须获得超级管理员的授权.其中,$G_1$ 代表仅与两个用户有业务关系且其中一个用户是“刘刚”的用户组;$G_2$ 组代表获得管理员授权的用户,并且该组用户要求与之有业务关系的用户的薪水高于 10 000;$G_0$ 组代表与 $G_1$ 组和 $G_2$ 组的用户均存在业务关系的非管理人员.$G_0$ 形式地表示为 $G_0=\neg A\cap\exists S.(\exists S.\{o\}\cap$

© 中国科学院软件研究所 http://www.jos.org.cn

$\leqslant 2S\cap\geqslant 2S)\cap\exists S.(\forall S^{-}.(\exists F.d)\cap\exists R.A\cap\forall P.(\exists R.B))$;(其中,$A$:Manager;$B$:Administrator;$R$:Granted-from;$P$:Managed-by;$S$:Have-business-relations-with;$F$:Salary;$d$:$\geqslant$10000;$o$:"LiuGang";$R\sqsubseteq P$;$Trans(P)=true$).我们计算 $G_0$ 的 CDNF 群及其可满足性(删除线表示备用子句).

为了尽可能地不让其他描述影响当前节点操作的直观效果,概念表述中会尽可能地用一些中间概念(见 $C_1$~$C_3$),同时用[]及下划线标记当前处理点描述,描述之后接着下一步操作说明(没有涉及到的步骤则省略),整个过程如下:

中间概念:$C_1=\exists S.\{o\}\cap\leqslant 2S\cap\geqslant 2S$;$C_2=\forall S^{-}.(\exists F.d)\cap\exists R.A\cap\forall P.(\exists R.B)$;$C_3=\exists R.B$.

初始:预处理阶段置 $Des(o)=\top$,因为 $G_0$ 已经是否定范式了,所以直接置 $G_0$ 为处理点 $P$:

$[\underline{\neg A\cap\exists S.C_1\cap\exists S.C_2}]$;——$P$ 重定位——>

$\neg A\cap\exists S.[\underline{\exists S.\{o\}\cap\leqslant 2S\cap\geqslant 2S}]\cap\exists S.C_2$;——数量限制满足:$\geqslant 2S$——>

$\neg A\cap\exists S.[\underline{\exists S.\{o\}\cap\leqslant 2S\cap\geqslant 2S\cap\exists S.\top\cap\exists S.\top}]\cap\exists S.C_2$;——数量限制满足:$\leqslant 2S$——>

$\neg A\cap\exists S.[\underline{\exists S.\{o\}\cap\leqslant 2S\cap\geqslant 2S\cap\exists S.\top}]\cap\exists S.C_2$;——$P$ 重定位——>

$\neg A\cap\exists S.(\exists S.[\underline{\{o\}}]\cap\leqslant 2S\cap\geqslant 2S\cap\exists S.\top)\cap\exists S.C_2$;——$P$ 重定位——>

$\neg A\cap\exists S.(C_1\cap\exists S.\top)\cap\exists S.[\underline{\forall S^{-}.(\exists F.d)\cap\exists R.A\cap\forall P.C_3}]$;——逆$\forall$处置:$\forall S^{-}.(\exists F.d)$——>

$[\underline{\neg A\cap\exists S.(C_1\cap\exists S.\top)\cap\exists S.(\forall S^{-}.(\exists F.d)\cap\exists R.A\cap\forall P.C_3)\cap\exists F.d}]$;——$P$ 重定位——>

$\neg A\cap\exists S.(C_1\cap\exists S.\top)\cap\exists S.[\underline{\forall S^{-}.(\exists F.d)\cap\exists R.A\cap\forall P.C_3}]\cap\exists F.d$;——$\forall$处置:$\forall P.C_3$——>

$\neg A\cap\exists S.(C_1\cap\exists S.\top)\cap\exists S.[\underline{\forall S^{-}.(\exists F.d)\cap\exists R.(A\cap\exists R.B\cap\forall P.C_3)\cap\forall P.C_3}]\cap\exists F.d$;——$P$ 重定位——>

$\neg A\cap\exists S.(C_1\cap\exists S.\top)\cap\exists S.(\forall S^{-}.(\exists F.d)\cap\exists R.[\underline{A\cap\exists R.B\cap\forall P.C_3}]\cap\forall P.C_3)\cap\exists F.d$;——$\forall$处置:$\forall P.C_3$——>

$\neg A\cap\exists S.(C_1\cap\exists S.\top)\cap\exists S.(\forall S^{-}.(\exists F.d)\cap\exists R.[\underline{A\cap\exists R.(B\cap\exists R.B\cap\forall P.C_3)\cap\forall P.C_3}]\cap\forall P.C_3)\cap\exists F.d$;——$P$ 重定位——>

$\neg A\cap\exists S.(C_1\cap\exists S.\top)\cap\exists S.(\forall S^{-}.(\exists F.d)\cap\exists R.(A\cap\exists R.[\underline{B\cap\exists R.B\cap\forall P.C_3}]\cap\forall P.C_3)\cap\forall P.C_3)\cap\exists F.d$;——$\forall$处置:$\forall P.C_3$——>

$\neg A\cap\exists S.(C_1\cap\exists S.\top)\cap\exists S.(\forall S^{-}.(\exists F.d)\cap\exists R.(A\cap\exists R.(B\cap\exists R.[\underline{B\cap\exists R.(B\cap\exists R.B\cap\forall P.C_3)\cap\forall P.C_3}]\cap\forall P.C_3)\cap\forall P.C_3)\cap\forall P.C_3)\cap\exists F.d$;——$P$ 重定位,并重用子句定位成功——>

$\neg A\cap\exists S.(C_1\cap\exists S.\top)\cap\exists S.(\forall S^{-}.(\exists F.d)\cap\exists R.(A\cap\exists R.(B\cap\exists R.(B\cap\exists R.[\sout{B\cap\exists R.B\cap\forall P.C_3}]\cap\forall P.C_3)\cap\forall P.C_3)\cap\forall P.C_3)\cap\forall P.C_3)\cap\exists F.d$.

最终得到一个完整的 CDNF 群:

$G_0'=\neg A\cap\exists S.(\exists S.\{o\}\cap\leqslant 2S\cap\geqslant 2S\cap\exists S.\top)\cap\exists S.(\forall S^{-}.(\exists F.d)\cap\exists R.(A\cap\exists R.(B\cap\exists R.(B\cap\exists R.(\sout{B\cap\exists R.B\cap\forall P.(\exists R.B)})\cap\forall P.(\exists R.B))\cap\forall P.(\exists R.B))\cap\forall P.(\exists R.B))\cap\exists F.d$;$Des(o)=\top$.根据此 CDNF 群,我们得出 $G_0$ 是可满足的.

### 2.3 CDNF群的森林结构及语义解释

在角色分层、数量限制以及逆算子的共同作用下,$\mathcal{SHOIN}(D)$-概念的语义解释已经失去了模型有限性[18].一个概念描述经过 CDNF 算法过程后,如果形成了可满足的完全析取范式群,则根据该范式群可以产生满足该概念的语义模型.

每个 CDNF 以角色为边,以处理点(对应于不同层的析取范式)为节点可以展开成明显的树型结构(不考虑重用边),我们称其为 CDNF 树.具体操作是:将 CDNF 中的所有处理点所对应子句中的$\exists$角色文字写成分离形式,即对于$\exists R.C$,写成$\exists R.\rightarrow C$,其中$\rightarrow$代表一条从角色指向限定概念的有向边.一个 CDNF 树自然就形成了.于是,整个范式群就可以形成一个 CDNF 森林了.因为每个节点都对应于一个子句,所以原来针对子句的称谓,照样适用

© 中国科学院软件研究所 http://www.jos.org.cn

于节点,如节点的前趋、后继,对于备用子句,相应地称为备用节点.为了便于生成一个语义模型,每个节点都赋予一个代号(但是这些代号不属于 CDNF 森林的必需部分),对于含具体数据类型(如 $d$)的节点(称为具体节点),则赋予一个满足该类型的值(表示为 $val(d)$)即可.

继续第 2.2 节的例子,$G_0$ 经过 CDNF 算法后,形成两个完全析取范式.图 2 中显示了 $G_0'$ 及 $Des(o)$对应的 CDNF 森林.

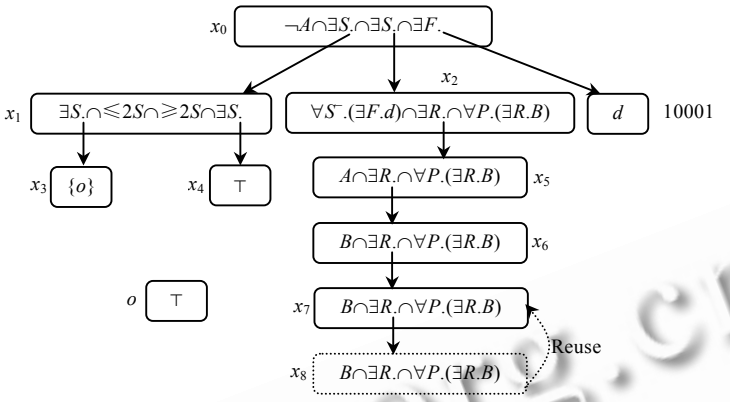


Fig.2 CDNF forest corresponding to concept $G_0'$ and $Des(o)$

图 2 概念$G_0'$及 $Des(o)$对应的 CDNF 森林

有了 CDNF 森林后,就可以构造出该 CDNF 森林所对应概念的一个模型.直观来说,我们这样定义一个解释域$\Delta^I$中的个体:$\Delta^I$中的一个个体对应于 CDNF 森林中一条从主 CDNF 根节点开始到某个非备用节点结束的通路.为了获得一个可能无限的解释域,这些通路可以是循环的,即一条通路并不一定碰到备用节点后就终止,而是可以不限次数地返回到对应的重用节点,并最终终止于某个非备用节点上.所以,一旦有重用发生,我们就可以获得一个无限的$\Delta^I$.

严格而言,假设 $T$ 是一棵无冲突的 CDNF 树.我们定义一个映射 $Tail(p)$来返回通路 $p$ 中的最后一个节点:给定通路 $p=[x_0,x_1,\ldots,x_n]$,$x_i$ 是树 $T$ 中的节点(均为非具体节点),$Tail(p)=x_n$.$T$ 中的通路归纳地定义如下:

1. 对于 $T$ 中的根节点 $x_0$,$[x_0]$是 $T$ 中的一条通路.
2. 对于 $T$ 中的一个通路 $p$ 和一个节点 $x_i$,$[p,x_i]$是 $T$ 中的一条通路,当且仅当:
   (a) $x_i$ 是 $Tail(p)$的一个后继,并且 $x_i$ 不是备用节点;
   (b) 存在 $T$ 中的某个备用节点 $y$,$y$ 是 $Tail(p)$的一个后继,并且 $x_i$ 被 $y$ 重用.

假设 $R$,$F$ 是 CDNF 群中出现的抽象及具体角色名,$A$ 是原始概念.$C'$是主 CDNF,则 $C'$的模型 $I=(\Delta^I,\cdot^I)$可以构造如下:

$\Delta^I=\{x_p|\ p$ 是主 CDNF 中的一个通路$\}$; $A^I=\{x_p|Tail(p)$中含有文字 $A\}$;

$\Phi(R)=\{\langle x_p,x_q\rangle\in\Delta^I\times\Delta^I|$要么 $q=[p,Tail(q)]$,且(1) $Tail(q)$是 $Tail(p)$的 $R$-后继,或

(2) 存在 $T$ 中的某个节点 $y$,$y$ 是 $Tail(p)$的 $R$-后继,并且 $Tail(q)$被 $y$ 重用;

要么 $p=[q,Tail(p)]$,且(1) $Tail(p)$是 $Tail(q)$的 $Inv(R)$-后继,或

(2) 存在 $T$ 中的某个节点 $y$,$y$ 是 $Tail(q)$的 $Inv(R)$-后继,并且 $Tail(p)$被 $y$ 重用$\}$;

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

$F^I=\{\langle x_p,val(d)\rangle|$边$\langle Tail(p),val(d)\rangle$上标记为 $F\}$;

© 中国科学院软件研究所 http://www.jos.org.cn

$C'^I=\{x_p|Tail(p)$是主 CDNF 树根节点$\}$.

现在我们来证明 $I$ 是概念 $C'$的模型.实际上,我们后面会证明,$I$ 当然也是初始输入概念的模型.

**引理 1**. 如果文字 $X\in cla(x_p)$($cla(x_p)$表示 $Tail(p)$所对应的节点子句),则 $x_p\in X^I$.

(1) 如果 $X$ 是原始概念 $A$,则根据 $I$ 的构造定义,显然有 $x_p\in X^I$.

下面我们对子句结构进行归纳,有:

(2) 如果 $X$ 是原始概念的非 $\neg A$,则根据 CDNF 的构造方式,则 $A\notin cla(x_p)$,所以 $x_p\in \Delta^I\backslash A^I=X^I$.

(3) 如果 $X$ 是⊤并且 $cla(x)$只含文字⊤,根据⊤的语义,显然有 $x_p\in \Delta^I=X^I$.

(4) 如果 $X$ 为$\leqslant nS$ 或$\geqslant nS$,则根据 CDNF 算法及 CDNF 树的构造过程,肯定有$\#\{\langle x_p,x_q\rangle|\langle x_p,x_q\rangle\in S^I\}\leqslant n$ 或$\geqslant n$,所以 $x_p\in X^I$.

(5) 如果 $X$ 为$\forall S.C$,如 $cla(x_p)$含有$\exists R$ 角色文字($R\sqsubseteq S$),则因为在 CDNF 构造过程中已经把 $C$ 通过$\forall$-Merge 操作结合到 $cla(x_p)$的每个$\exists R$ 角色文字中去了,所以由这些文字的限定概念所对应的个体都必然是概念 $C$ 的实例,所以 $x_p\in X^I$;而如果 $cla(x_p)$不含有$\exists R$ 角色文字,则没有个体与 $cla(x_p)$有 $S$ 关系,显然有 $x_p\in(\forall S.C)^I=X^I$;类似的道理同样适应于$(\forall F.d)$.

(6) 如果 $X$ 为$\exists R.C$,则根据 CDNF 树构造方法,显然存在一个个体 $x_q\in(C)^I$ 满足:$\langle x_p,x_q\rangle\in(R)^I$,所以 $x_p\in X^I$.类似的道理同样适应于$(\exists F.d)$.

**命题 1**. $I$ 是概念 $C'$的模型.

证明:概念 $C'$实际上就是 $C'$顶层析取范式所对应节点(如 $x_0$)所表示的概念,根据引理 1,通路$[x_0]$属于每一个文字.在这些文字中,只有同角色的$\exists/\forall$相互是不独立的,但在经过 CDNF 算法过程后,$\forall$文字对$\exists$文字的作用已经完成,实际上是已不再需要了.所以除了$\forall$文字以外,其他文字皆相互独立,因而$[x_0]$也必然属于这些文字的$\cap$组合,所以有$[x_0]\in(C')^I$,即 $I$ 是 $C'$的模型. □

根据图 2,我们可以很容易地找到一个 $G_0'$的模型(当然也是 $G_0$ 的模型).这里给出的模型是有限的($\mathcal{SHOIN}(D)$-的语义解释并非必须是无限的),还可以另外根据上面所述方法找出一个无限模型.为了便于描述,我们把通路$[x_0,x_1,x_2,\ldots]$简写成 $x_{012,\ldots}$:

$\Delta^I=\{x_0,x_{01},x_{013}(=o),x_{014},x_{02},x_{025},x_{0256},x_{02567}\}$;

$R^I=\{\langle x_{02},x_{025}\rangle,\langle x_{025},x_{0256}\rangle,\langle x_{0256},x_{02567}\rangle,\langle x_{02567},x_{02567}\rangle\}$;$S^I=\{\langle x_0,x_{01}\rangle,\langle x_0,x_{02}\rangle,\langle x_{01},x_{013}\rangle,\langle x_{01},x_{014}\rangle\}$;$P^I=\{\langle x_{02},x_{025}\rangle,\langle x_{025},x_{0256}\rangle,\langle x_{0256},x_{02567}\rangle,\langle x_{02567},x_{02567}\rangle,\langle x_{02},x_{0256}\rangle,\langle x_{02},x_{02567}\rangle,\langle x_{025},x_{02567}\rangle\}$;$F^I=\{\langle x_0,10001\rangle\}$;

$A^I=\{x_{025}\}$;$B^I=\{x_{0256},x_{02567}\}$;$G_0^I=G_0'^I=\{x_0\}$.

### 2.4 CDNF算法的可靠性、完备性和终止性

本节先给出 4 个引理,证明(逆)$\forall$-Merge,(逆)$\exists$-Merge,$o$-Merge 操作和数量限制操作的语义特性,然后在此基础上证明 CDNF 算法的可靠性、完备性和终止性.

**引理 2**. 一个概念 $G$ 在应用一次逆$\forall$-Merge 操作后形成新的概念 $G'$,则 $G$ 与 $G'$具有模型一致性,即如果 $I$ 是 $G$ 的模型,则 $I$ 也是 $G'$的模型,反之亦然.

证明:设 $G=\exists R.(C_2\cap\forall S^-.C_3)\cap C_1$($C_i$为 $G$ 子概念,$R\sqsubseteq S$),因为 $R,S,C_i$的普遍性,所以 $G$ 可以代表各种需要执行逆$\forall$-Merge 操作的概念,对 $G$ 应用逆$\forall$-Merge 操作后,得到 $G'=\exists R.(C_2\cap\forall S^-.C_3)\cap C_1\cap C_3$(对于 $Trans(R)=false$)或 $G'=\exists R.(C_2\cap\forall S^-.C_3)\cap C_1\cap C_3\cap\forall R^-.C_3$(对于 $Trans(R)=true$).

假设 $I$ 是 $G$ 的模型,则必然存在 $a\in\Delta^I$,有 $a\in G^I$,如此则存在 $b\in\Delta^I$,有$\langle a,b\rangle\in R^I$,并且 $b\in(C_2\cap\forall S^-.C_3)^I$,也即$\langle b,a\rangle\in(R^-)^I\sqsubseteq(S^-)^I$,所以必须有 $a\in C_3^I$,所以也就有 $a\in(G\cap C_3)^I$.如果 $Trans(R)=true$,假设还存在 $c\in\Delta^I$,有$\langle a,c\rangle\in(R^-)^I$并且 $c\notin C_3^I$,则根据 $R$ 的传递性有$\langle b,c\rangle\in(R^-)^I$,可得 $c\in C_3^I$.由此矛盾.因此应有 $a\in(G\cap C_3\cap\forall R^-.C_3)^I$,所以不管 $R$ 是否传递,$I$ 都是 $G'$的模型.

反之,设 $I$ 是 $G'$的模型,则必然存在 $a\in\Delta^I$,根据 $R$ 的传递性,有 $a\in(G')^I=(G\cap C_3)^I$或 $a\in(G')^I=(G\cap C_3\cap\forall R^-.C_3)^I$,所以 $I$ 也是 $G$ 的模型.

© 中国科学院软件研究所 http://www.jos.org.cn

所以,一个概念描述在应用逆∀-Merge 操作后,并不会导致概念的模型发生变化,即模型等价,这种操作我们称为模型保持操作. □

**引理 3**. ∀-Merge 操作是模型保持操作.

证明:因为每次∀-Merge 操作只影响两个角色文字,所以我们设 $G=\exists R.d_1\cap\forall S.d_2\cap C_3$(其中,$S,R$ 是具体角色,$R\sqsubseteq S$,$d_1$ 和 $d_2$ 是数据类型),那么在应用∀-Merge 操作后,$G'=\exists R.(d_1\cap d_2)\cap\forall S.d_2\cap C_3$.

假设 $I$ 是 $G$ 的模型,则必然存在 $a\in\Delta^I$,有 $a\in G^I$,因为文字$\exists R.d_1$ 的存在,则必存在值 $t\in\Delta^D$,有$\langle a,t\rangle\in R^I\sqsubseteq S^I$,并且 $t\in d_1^D$,又因为文字$\forall S.d_2$ 的存在,所以 $t\in d_1^D$.所以也就有 $a\in(\exists R.(d_1\cap d_2)\cap\forall S.d_2\cap C_3)^I=(G')^I$,所以 $I$ 也是 $G'$的模型.

反之,设 $I$ 是 $G'$的模型,因为$(d_1\cap d_2)\subseteq d_1$,所以根据语义有$\exists R.(d_1\cap d_2)\subseteq\exists R.(d_1)$,更进一步,有:$\exists R.(d_1\cap d_2)\cap\forall S.d_2\cap C_3\subseteq\exists R.d_1\cap\forall S.d_2\cap C_3$,即 $G'\subseteq G$,所以 $I$ 也是 $G$ 的模型. □

同样的证明思路结合引理 2 中关于传递性角色的讨论可得出,对于抽象角色的∀-Merge 操作也满足模型保持的特点.

**引理 4**. 假设概念 $G$ 在经过 CDNF 算法的≥满足操作后产生的新概念为 $G'$.那么,$G$ 是可满足的,当且仅当 $G'$是可满足.

证明:(1) “当”方向:如果概念 $G'$是可满足的,则 $G$ 是可满足的.

假设文字$\geqslant nS$ 是 $G$ 中当前处理子句 $x$ 的文字,并对其应用≥满足操作后得到 $G'$,相应的 $x$ 子句变成了 $x'$子句,并且 $x'\subseteq x$.假设 $I$ 是 $G'$的模型,则必然存在 $a\in\Delta^I$,有 $a\in(x')^I$,所以也就有 $a\in(x)^I$.而除 $x$ 子句外,$G$ 与 $G'$其他部分完全相同,所以 $I$ 也是 $G$ 的模型.

(2) “仅当”方向:如果 $G$ 是可满足的,则 $G'$是可满足的.

假设算法的当前处理点为概念 $G$ 中的某个子句 $x$(含文字$\geqslant nS$)经≥满足操作后变成为 $x'=x\cap\exists S.C_1\cap\exists S.C_2\cap\ldots\cap\exists S.C_n(C_i=\top,1\leqslant i\leqslant n)$,相应地,$G$ 变成 $G'$.$I=(\Delta^I,\cdot^I)$是 $G$ 的模型.所以应该有一个个体 $a\in\Delta^I$满足 $a\in(x)^I$,并且至少存在 $c_1,c_2,\ldots,c_n\in\Delta^I$,使得$(a,c_i)\in S^I$.把这 $n$ 个 $c_i$ 分摊到新产生的 $n$ 个$\exists S.C_i(C_i=\top,1\leqslant i\leqslant n)$,即 $c_i\in(C_i)^I=(\top)^I$,这是显然的.由此可见,$a\in(x')^I$.所以≥满足也是模型保持操作. □

**引理 5**. $o$-Merge 操作是语义等价操作.

证明:$o$-Merge 操作只是将一个子句的整体概念从原来的位置移到了孤点标记上.假设当前某子句 $x=\{o\}\cap C_1$,而孤点 $Des(o)$当前标记为 $C_2$.因为 $o$ 是命名个体,所以在语义上有:$o\in(x)^I$,即有 $o\in(C_1)^I$.而因为 $C_2$ 为孤点标记概念,所以同时有 $o\in(C_2)^I$.所以,个体 $o$ 实际上必须满足:$o\in(C_1\cap C_2)^I$.而这正是 $o$-Merge 操作后的结果:$x'=\{o\}$,孤点 $Des(o)$标记为 $C_1\cap C_2$.反之亦然,即这个过程是完全可逆的.也就是说,一个 CDNF 群在经历一次 $o$-Merge 操作后,在语义上没有发生任何变化,所以称其为语义等价操作. □

**引理 6**. 假设概念 $G$ 在经过 CDNF 算法的≤满足操作后产生的新概念为$\{G_1',G_2',\ldots,G_n'\}$.那么,$G$ 是可满足的,当且仅当至少 1 个新概念$G_i'(1\leqslant i\leqslant n)$是可满足.

证明:(1) “当”方向:如果概念$G_i'(1\leqslant i\leqslant n)$是可满足的,则 $G$ 是可满足的.

注意到公式:$C_j\cap\exists(R^-,S^-).(C_i)\subseteq\exists R^-.(C_i\cap\exists S.C_j)$(对应于逆∃-Merge 操作)以及$\exists(R,S).(C_i\cap C_j)\subseteq\exists S.C_i\cap\exists R.C_j$(对应于∃-Merge 操作)对任何简单角色名 $S,R$ 以及概念 $C_i,C_j$ 都是成立的.由此可知,(逆)∃-Merge 操作实际上是外延收缩的操作.所以只要某个$G_i'(1\leqslant i\leqslant n)$是可满足的,则 $G$ 也是可满足的.

(2) “仅当”方向:如果 $G$ 是可满足的,则至少 1 个新概念$G_i'(1\leqslant i\leqslant n)$是可满足的.

假设算法的当前处理点为概念 $G$ 中的某个子句 $x=\exists R_1.C_1\cap\exists R_2.C_2\cap\ldots\cap\exists R_m.C_m\cap\leqslant nS\cap\ldots(n<m$ 且对于 $1\leqslant i\leqslant m$ 有 $R_i\sqsubseteq S)$,$I=(\Delta^I,\cdot^I)$是 $G$ 的模型.则至少有 1 个个体 $a\in\Delta^I$满足 $a\in x^I$,并且存在 $c_1,c_2,\ldots,c_m\in\Delta^I$,使得$(a,c_i)\in(R_i)^I\sqsubseteq S^I$且 $c_i\in(C_i)^I$.为了满足≤数量限制,则肯定要有一些 $c_i$ 为同一个体.根据这一事实,我们对 $G$ 进行如下的≤满足操作:如果某 $c_i$ 是 $a$ 的前趋,则对$\exists R_i.C_i$ 实现逆∃-Merge 操作;否则,如果 $c_i$ 与某 $c_j$ 为同一个体,则对$\exists R_j.C_j$,$\exists R_i.C_i$ 实现∃-Merge 操作.

经过上述过程,我们可以构建出一个新的子句 $x'$,相应地,$G$ 变成了 $G'$.解释 $I$ 显然也是 $G'$的模型.这种根据相关的信息来指导执行∃-Merge 的操作,我们称其为有指导的操作. □

© 中国科学院软件研究所 http://www.jos.org.cn

**命题 2(可靠性与完备性)**. 一个概念 $G$ 是可满足的,当且仅当至少 1 个由它产生的主 CDNF 也是可满足的,并且这些可满足的主 CDNF 的模型也肯定是原概念 $G$ 的模型.

证明:假设概念 $G$ 经 CDNF 算法过程后形成了主 CDNF: $G'_1,G'_2,...,G'_n$.

(1) "当"方向,即可靠性.由引理 2~引理 6 可知,(逆)∀-Merge,≥满足操作是属于模型保持操作,(逆)∃-Merge 操作则是外延收缩操作.$o$-Merge 操作以及形成 CDNF 的过程根据∩,∪的语义可知是在语法与语义上均等价的操作.而 CDNF 形成之后取其中之一子句又是外延收缩的操作.所以肯定有:$G'_i \subseteq G(1 \leqslant i \leqslant n)$成立.如果某个主 CDNF 是可满足的,比如说$G'_i(1 \leqslant i \leqslant n)$.假设 $\mathcal{I}=(\Delta^{\mathcal{I}},\cdot^{\mathcal{I}})$是$G'_i$的模型,那么 $I$ 也是 $G$ 的模型.当然,也可能存在 $G$ 中出现的原始概念或角色名并不出现在$G'_i$中的情况,$\cdot^{\mathcal{I}}$只需将它们全部映射为∅即可,而且这并不影响到 $I$ 是 $G$ 的模型这一特性.

(2) "仅当"方向,即完备性.假设 $\mathcal{I}=(\Delta^{\mathcal{I}},\cdot^{\mathcal{I}})$是 $G$ 的模型.从 $G$ 开始,我们可以生成一个 $G$ 的主 CDNF,如 $G'$,使得 $I$ 也是 $G'$的模型.为了确保产生出来的 $G'$具有 $I$ 是其模型的特性,要对 CDNF 算法作如下简单的修改:

(1) 对每个处理点 $C$,根据语义,肯定有个个体 $a \in \Delta^{\mathcal{I}}$,使得:$a \in C^{\mathcal{I}}$.在 $C$ 形成析取范式后,如果产生多个子句,如 $C'_1.C'_2,....C'_m$,则必然存在 $i(1 \leqslant i \leqslant m)$,使得 $a \in C_i^{\mathcal{I}}$ 成立.在进行其他操作之前选择子句 $C_i$ 进行操作、展开,其他子句全部删除.

(2) 在对子句进行≤满足操作时,按照引理 6 中描述的方式去做,实行有指导的≤满足操作.

这样,我们保证对于 CDNF 算法每一步操作产生的新的描述,$I$ 都是它的模型.所以最终产生出来的 $G'$,$I$ 当然也是它的模型. □

**引理 7**. 设 $m$ 为概念 $G$ 的子概念数,$n>2^{3m}$.设 $x_1,x_2,\ldots,x_n$,是概念 $G$ 的 CDNF 群中的某个 CDNF 中的一条路径,则存在一个 $x_i$ 为可被其某个子孙 $x_j$ 重用.

证明:一个 $G$ 的子概念最多只能有 $m$ 种可能,这样,以下 3 个不等式成立:(1) $|\{\mathcal{R}(x_i.Letter)\times\mathcal{E}(x_i.Letter)|2 \leqslant i \leqslant n\}| \leqslant 2^m$;(2) $|\{\mathcal{U}pper(x_i)-x_i.Letter|2 \leqslant i \leqslant n\}| \leqslant 2^m$;(3) $|\{x_i/\mathcal{R}_\cdot|2 \leqslant i \leqslant n\}| \leqslant 2^m$.所以,在 $n$ 个子句组成的路径中,必然存在有两个子句 $x_i,x_j(j>i)$满足:

$\mathcal{R}(x_i.Letter)=\mathcal{R}(x_j.Letter)$,$\mathcal{E}(x_i.Letter)=\mathcal{E}(x_j.Letter)$,$\mathcal{U}pper(x_i)-x_i.Letter=\mathcal{U}pper(x_j)-x_j.Letter$,并且 $x_i/\mathcal{R}=x_j/\mathcal{R}$,这就意味着 $x_i$ 为可被 $x_j$ 重用的子句. □

**命题 3(终止性)**. CDNF 算法在产生并处理一定数量的处理点后必然终止.

证明:设 $m$ 为概念 $G$ 的子概念数,显然 $m$ 是线性于 $G$ 的长度的.CDNF 算法的终止性由下面 3 条 CDNF 算法的性质决定.

(1) CDNF 算法运行过程中不会移去处理点,除了 $o$-Merge 和(逆)∃-Merge 操作以外,也不会从处理点的子句中移去文字.

(2) 新的处理点只有$\exists R.C$ 型及$\geqslant nS.C$ 文字产生.$\exists R.C$ 型文字至多产生一个处理点,$\geqslant nS.C$ 文字则最多产生 $n$ 个处理点,而这种文字最多有 $m$ 种.所以形成的处理点出度最多为 $nm$.

(3) 根据引理 7,概念 $G$ 的 CDNF 的路径长度最长为 $2^{3m}$. □

### 2.5 扩展到一般包含公理

我们前面已经讲述了在没有 *TBox* 或非循环 *TBox* 情况下的概念可满足性问题.但是,如果允许出现形式为 $C \subseteq D$(这里 $C$ 和 $D$ 可能是复杂的描述)的一般包含公理,展开就不再可能了.其实,只需考虑单个公理$\top \subseteq \hat{C}$(这里 $\hat{C}=(\neg C_1 \cup D_1) \cap \ldots \cap (\neg C_n \cup D_n)$)就足够了[13],而不用去考虑有限多个 $C_1 \subseteq D_1,\ldots,C_n \subseteq D_n$ 公理.公理$\top \subseteq \hat{C}$ 说明任何个体都必须属于概念 $\hat{C}$.上面介绍的 CDNF 算法可以稍作修改,从而把这个公理考虑进去:所有节点的概念描述在首次生成析取范式前都与 $\hat{C}$ 进行∩操作,然后基于新的描述去生成析取范式即可.但是,这样修改可能导致文字等长地向下传递.子句重用技术依然可以解决这个问题,所以算法并不需要修改.

### 2.6 复杂性问题讨论

$\mathcal{SHOIN}(D)$的概念可满足性问题是 NExpTime 完全问题[19].因为 CDNF 算法的时间和空间代价都集中体现

© 中国科学院软件研究所 http://www.jos.org.cn

在处理点个数上,而处理点的个数就是输入概念经 CDNF 算法过程最终产生的各级∃文字的个数+1.这个个数在大多数情况下是线性于输入概念描述的长度.但在有些情况下,它却可达指数级,因为∀文字和∃文字的相互作用就可能产生指数级数量的∃文字,如本例[13]:$C_1=\exists R.A\cap\exists R.B$;…;$C_n=\exists R.A\cap\exists R.B\cap\forall R.C_{n-1}$.这种情况下,对 $C_n$ 作用产生出来的 CDNF 的长度是可能达到指数于输入概念描述 $C_n$ 长度的.此外,更为主要的原因是,由引理 7 可知,因为传递性∀角色文字的长度不减型的向下传递,导致 CDNF 树的路径长度最长可达 $2^{3m}$($m$ 为输入概念的子概念数),所以这是导致 CDNF 算法只能止步于指数级时间空间的最关键因素.

## 3 CDNF 与 Tableau 之间的性能比较

在描述逻辑中,稍为复杂的语言系统基本上都采用 Tableau 算法来判断概念的可满足性.所以我们在这里就这两种算法在时间和空间性能上作一比较.从比较中我们可以得出,CDNF 算法在时间/空间上比 Tableau 算法具有更好的效率.

与 Tableau 算法相比,CDNF 算法对∩,∪,∃算子进行了不同的处理.对∃算子的处理不同之处仅在于:生成一个节点时,Tableau 算法要把限定概念复制到新的节点上,而 CDNF 则相当于直接在原来的限定概念上进行处理.而对∩,∪算子的处理都是属于节点内部的处理,即∩,∪算子不会导致节点的产生或减少.这样一来,这两种算法的区别也就体现在节点之内的处理方式上了.事实上,也正因为如此,以 CDNF 和 Tableau 算法对同一概念进行处理,产生模型是一致的.

就一个节点而言,Tableau 算法利用∩/∪-规则对概念描述进行层层展开,而 CDNF 算法则将其组装成一个析取范式并取其中一个子句.例如,对某节点初始描述 $x_0=\{A\cap B\cap(C\cup B)\}$,由 Tableau 算法可得 $x_0=\{A\cap B\cap(C\cup D),A,B\cap(C\cup D),B,(C\cup D),C/D\}$,而 CDNF 算法的结果则是,$x_0=\{A\cap B\cap C/D\}$.假设一个节点内的概念描述长度为 $n$,则为了展开这段描述,Tableau 算法最多要扫描 $n$ 次,每次长度最多为 $n$,所以其时间和空间复杂度均为 $O(n^2)$.而 CDNF 算法只需扫描 1 次即可生成析取范式并获得其中一个子句,所以其时间和空间复杂度均为 $O(n)$.这是 CDNF 算法优越性的第 1 个体现.

另外,在对∃的处理上,正如本节之前所述,Tableau 算法以复制形式产生新节点描述,而 CDNF 则直接在原限定概念上处理.例如,对于节点$(\exists R.C)(x)$,Tableau 算法产生一个新节点和边:$C(y),R(x,y)$.而 CDNF 算法则直接对$(\exists R.C)$中的 $C$ 进行处理的.这是 CDNF 算法优越性的第 2 个体现.当然,为了防止传递角色可能导致的不可终止性,CDNF 保留传递性角色的∃文字的原始限定概念.但是,传递性角色在实际应用中只占很小的一部分,所以 CDNF 在这一方面节省的空间仍然是可观的.

这样一来,由于节点的数量可能线性于或指数于输入概念描述长度,CDNF 算法相对于 Tableau,就可以节省线性于或指数于输入概念长度的空间和时间代价了.仍然以第 2.2 节的概念 $G_0$ 为例,我们可以看一下采用一般 Tabeau 算法所消耗的空间(如图 3 所示).对比图 2 和图 3,我们对两个图的节点中的概念描述进行统计得出($G_0$ 的描述长度是 53 字符),Tableau 森林所占空间为 321 字符,是 $G_0$ 长度的 6 倍,而 CDNF 森林所占空间仅为 120 字符,为 $G_0$ 长度的 2.2 倍.由于∩,∪,∃算子几乎在所有概念描述中出现的比例都很大,所以这个例子所体现出的性能差异也就带有相当的普遍性.在实践中,我们在自建的 Agent 通信本体语言库中对比了两种算法,所得的结果与上面的分析基本是一致的.

## 4 结论与讨论

Tableau 算法借助 *ABox* 的一致性来检测 $\mathcal{SHOIN}(D)$-概念的可满足性.而 CDNF 算法则直接在 $\mathcal{SHOIN}(D)$-概念描述上构建完全析取范式群,得出该概念可满足性的直接判断.Tableau 算法以个体名为核心进行展开,而 CDNF 则以子句为核心进行展开.二者应该说是相通的,因为子句在形成解释时正好对应一个个体名.为了防止不必要的扩展,Tableau 算法采用了成对阻塞[18],成对阻塞显然不适用于 CDNF 算法,因为 CDNF 中父节点是包含子节点的.于是 CDNF 采用了子句重用.二者有异曲同工之效.

在算法效率上,CDNF 是对概念可满足性的直接判断,尤其在对∩,∪,∃算子的处理上,CDNF 明显比 Tableau

© 中国科学院软件研究所 http://www.jos.org.cn

要好很多.相对而言,CDNF 算法可以节省从线性于到指数于输入概念长度的空间代价.当然,Tableau 算法还可用于其他推理任务,如实例检测等(虽然性能未必令人满意),而 CDNF 目前还只针对概念(关于 *TBox*,*RBox* 的)可满足性问题.

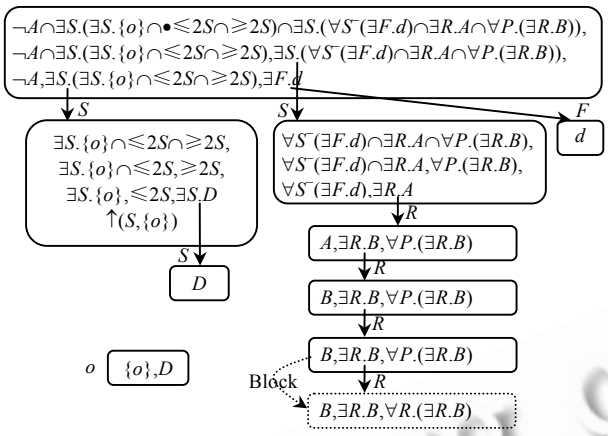


Fig.3 Tableau forest built from concept $G_0$

图 3 由概念 $G_0$ 构建出的 Tableau 森林

下一步对 CDNF 算法的研究重点应该放在两个方面:一是跟进基于 $\mathcal{SHOIN}(D)$[20]的研究,因为它是 OWL2 的逻辑基础,要着重解决复合角色包含问题,毕竟目前基于自动机的解决办法仍然不够实用;二是探索 CDNF 算法在其他推理方面的应用,如整个知识库的一致性、实例检测等,以更好地发挥其作用.

**References**:

[1] Fensel D, van Harmelen F, Horrocks I, McGuinness DL, Patel-Schneider PF. OIL: An ontology infrastructure for the semantic Web. IEEE Intelligent Systems, 2001,16(2):38−45.

[2] Connolly D, van Harmelen F, Horrocks I, McGuinness DL, Patel-Schneider PF, Stein LA. DAML+OIL (March 2001) reference description. W3C Note, 2001. http://www.w3.org/TR/2001/NOTE-daml+oil-reference-20011218

[3] van Harmelen F, Hendler J, Horrocks I, McGuinness DL, Patel-Schneider PF, Stein LA. OWL Web ontology language reference. W3C Working Draft, 2003. http://www.w3.org/TR/2003/WD-owl-ref-20030331

[4] Grau BC, Horrocks I, Motik B, Parsia B, Patel-Schneider P, Sattler U. OWL 2: The next step for OWL. Journal of Web Semantics: Science, Services and Agents on the World Wide Web, 2008,6(4):309−322.

[5] Hustadt U, Motik B, Sattler U. Reasoning in description logics by a reduction to disjunctive datalog. Journal of Automated Reasoning, 2007,39(3):351−384. [doi: 10.1007/s10817-007-9080-3]

[6] Halashek-Wiener C, Parsia B, Sirin E. Description logic reasoning with syntactic updates. In: Meersman, R, Tari Z, eds. Proc. of the 5th Int'l Conf. on Ontologies, Databases, and Applications of Semantics (ODBASE 2006). LNCS 4275, Berlin, Heidelberg: Springer-Verlag, 2006. 722−737.

[7] Horrocks I, Sattler U. A tableau decision procedure for SHOIQ. Journal of Automated Reasoning, 2007,39(3):249−276. [doi: 10.1007/s10817-007-9079-9]

[8] Lutz C, Miličić M. A tableau algorithm for description logics with concrete domains and general TBoxes. Journal of Automated Reasoning, 2007,38(1-3):227−259. [doi: 10.1007/s10817-006-9049-7]

[9] Chang L, Lin F, Shi ZZ. A dynamic description logic for representation and reasoning about actions. In: Zhang Z, Siekmann J, eds. Proc. of the KSEM 2007. LNAI 4798, Berlin, Heidelberg: Springer-Verlag, 2007. 115−127.

[10] Chang L, Shi ZZ, Qiu LR, Lin F. A tableau decision algorithm for dynamic description logic. Chinese Journal of Computers, 2008, 31(6):896−909 (in Chinese with English abstract).

© 中国科学院软件研究所 http://www.jos.org.cn

[11] Jiang YC, Shi ZZ, Tang Y, Wang J. Fuzzy description logic for semantics representation of the semantic Web. Journal of Software, 2007,18(6):1257−1269 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/18/1257.htm [doi: 10.1360/jos181257]

[12] Stoilos G, Stamou G, Pan JZ, Tzouvaras V, Horrocks I. Reasoning with Very Expressive Fuzzy Description Logics. Journal of Artificial Intelligence Research, 2007,30(1):273−320.

[13] Baader F, Calvanese D, McGuinness DL, Nardi D, Patel-Schneider PF. The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge: Cambridge University Press, 2003.

[14] Horrocks I, Pate1-Schneider PF, van Harmelen F. From SHIQ and RDF to OWL: The Making of a Web Ontology Language. Journa1 of Web Semantics, 2003,1(1):7−26.

[15] Baader F, Hanschke P. A scheme for integrating concrete domains into concept languages. In: Mylopoulos J, Reiter R, eds. Proc. of the IJCAI'91. San Fransisco: Morgan Kaufmann Publishers, 1991. 452−457.

[16] Schaerf A. Reasoning with individuals in concept languages. Data and Knowledge Engineering, 1994,13(2):141−176. [doi: 10.1016/0169-023X(94)90002-7]

[17] Horrocks I, Sattler U. Ontology reasoning in the $\mathcal{SHOQ}(\boldsymbol{D})$ description logic. In: Bernhard N, ed. Proc. of the IJCAI 2001. San Francisco: Morgan Kaufmann Publishers, 2001. 199−204.

[18] Horrocks I, Sattler U, Tobies S. Practical reasoning for expressive description logics. In: Ganzinger H, McAllester D, Voronkov A, eds. Proc. of the 6th Int'l Conf. on Logic for Programming and Automated Reasoning (LPAR'99). LNAI 1705, Berlin, Heidelberg: Springer-Verlag, 1999. 161−180.

[19] Tobies S. Complexity results and practical algorithms for logics in knowledge representation [Ph.D. Thesis]. LuFG Theoretical Computer Science, RWTH-Aachen, 2001.

[20] Horrocks I, Kutz O, Sattler U. The even more irresistible SROIQ. In: Doherty P, Mylopoulos J, Welty C, eds. Proc. of the 10th Int'l Conf. of Knowledge Representation and Reasoning (KR 2006). Menlo Park: AAAI Press, 2006. 57−67.

**附中文参考文献:**

[10] 常亮,史忠植,邱莉榕,林芬.动态描述逻辑的 Tableau 判定算法.计算机学报,2008,31(6):896−909.

[11] 蒋运承,史忠植,汤庸,王驹.面向语义 Web 语义表示的模糊描述逻辑.软件学报,2007,18(6):1257−1269. http://www.jos.org.cn/1000-9825/18/1257.htm [doi: 10.1360/jos181257]

**古华茂**(1975－),男,江西安远人,博士,讲师,主要研究领域为知识表示与推理,描述逻辑,Web 智能,流形学习.

**凌云**(1962－),男,教授,主要研究领域为智能信息处理,自组织网络.

**王勋**(1967－),男,博士,教授,CCF 高级会员,主要研究领域为智能信息处理,虚拟现实,移动图形计算.

**高济**(1946－),男,教授,博士生导师,主要研究领域为人工智能,自治计算,语义网格.

© 中国科学院软件研究所 http://www.jos.org.cn