南京航空航天大学

论 文 集

（一九九九年）

第 15 册

（八系）

南京航空航天大学科技部编

二〇〇〇年三月

八　　系

目　录

八〇一教研室(1篇)

八〇二教研室(5篇)

八〇三教研室(2篇)

八〇四教研室(1篇)

八〇五教研室(10篇)

AN APPROACH TO C SOFTWARE REUSE BASED ON DATABASE TECHNIQUES*

Qin Xiaolin *Lin Junhai*

(Department of Computer Science and Engineering, NUAA
29 Yudao Street, Nanjing 210016, P. R. China)

ABSTRACT

This paper presents a tool for managing, reusing and analysing C software code based on database techniques. The abstract information of entire software code is stored in a program database that is the conceptual scheme of the entire software, whereas the reuse component is a subscheme. Relational algebra can be conveniently used to manage, analyse and reuse C code. In the tool, we can manage, analyse and reuse any components in the program database and rapidly extract source code of any components or construct the program code of a new system. The rule system is introduced in reusing source code.

Key words: software reengineering; software reuse; program database

INTRODUCTION

The software reengineering is proposed in order to solve software crisis[1,2], which is to make use of the information of program code to implement software maintenance, reverse engineering, program comprehension and software reuse[1~5]. As we known, if a new software is developed based on an existing software, the cost of software development is reduced, software quality is improved and the period of software development is shortened. The software reuse is widely considered as a way to increase the productivity and improve the quality and reliability of new software systems. The paper mainly presents how to manage and reuse the program code of software systems.

It is crucial to efficiently store and manage the source code in order to make use of the program code. We design and implement a tool, C software reuse tool (CSRT), for managing, analysing and reusing C program code. In CSRT, the abstract information about program code, including any components (e. g. file or function) and reference relationships between them, is stored in a program database, which is the conceptual scheme of entire software system. Users can extract the subscheme of any components from the conceptual scheme and then generate the program code of the component. New components can also be added in the program database and then the program code of a new system is generated.

1 COMPONENTS AND PROGRAM DATABASES OF C CODE

1.1 Component analysis

There are five global objects in C program:

* Received 19 Mar. 1999; revision received 27 Apr. 1999

file, function, global variable, macro and type[6]. The directed graph can be used to show the call relationships between functions. The vertices represent functions; the directed edges represent reference relationships between functions; the dotted lines represent the binding relationships of the global variables, macros and types between functions; the dotted lines with a small triangle represent global variables, macros, types, and H files referenced by the function. Fig. 1 shows a reference relationship, in which the function f_2 and f_4 refer to the global variables x and y, whereas the global variable z is only referenced by the function f_4.

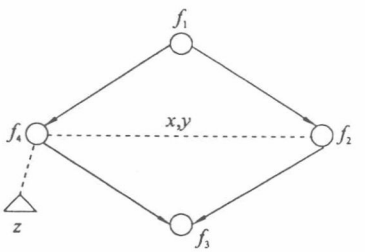

Fig. 1 A reference relationship

Let $f(H,F,V,M,T)$ represent the objects referenced by the function f. F is a set of functions which are called directly and indirectly by the function f; V,M,T and H are a set of global variables, a set of macros, a set of user defined-types, and a set of H files, which are referred by all functions in F respectively. Thus, for the function f_1, a complete component that can be compiled and executed should be

$$f_1(\Phi,\{f_1,f_2,f_3,f_4\},\{x,y,z\},\Phi,\Phi)$$

1.2 Program databases

It is key to clearly know the definition and reference information about five global objects if we would like to understand or reuse C programs. Therefore, the program database must include the definition and the reference information about files, functions, global variables, macros, and types. Meanwhile, it should entirely reflect the definition and the reference information about various software versions because a reusing component is sometimes extracted from different software versions. EXTERN and GOTO statements should be considered.

Based on Ref. [8], the program database consists of 21 relations. 6 relations are used to represent the definition information of 5 global objects in C programs. For instance, **funobj** (ver, funname, filename, funtype, function, static, bline, eline, environment, author, date) stores version number, function name, file name in which the function is, function type, what the function can do, begin line, end line, run environment, author and date respectively. If the attribute static is zero, the function is nonstatic; otherwise, it is static, which means that its scope is limited in the file filename. 12 relations represent the reference relationships between global objects. For example, **funfun** (ver, funname 1, funname 2, lineno, flag) represents that the function funname 1 refers to the function funname 2 at line lineno in the version ver. If the attribute flag is zero, it is a call; otherwise, it is an extern declaration. The relation **labobj** is used to store the definition and reference information about LABEL and GOTO statements. We also use 2 relations, version and lib, to store the information about software versions and function libraries.

2 SYSTEM DESIGN

Our aim is to design and implement a tool for managing and reusing C source code as C function libraries. In order to use the code of existing software efficiently, the abstract information of entire software code is stored into the program database that forms a friendly interface between users and program code. Based on the information in the program database, query language and some tools can be used to know the

definition and reference information of any C program object, such as file, function and variable in the software system. Meanwhile, the subscheme of the definition and reference information about any components can be easily created, on which the code of the component can be generated. Users can also directly call functions in the program database from new software to generate the entire code of the new software rapidly. Besides, based on the program database, the program analyser can do some analysis tasks, such as function call view, reference view between two objects, dead code analysis, binding analysis between objects. Fig. 2 shows the architecture and functions of CSRT.

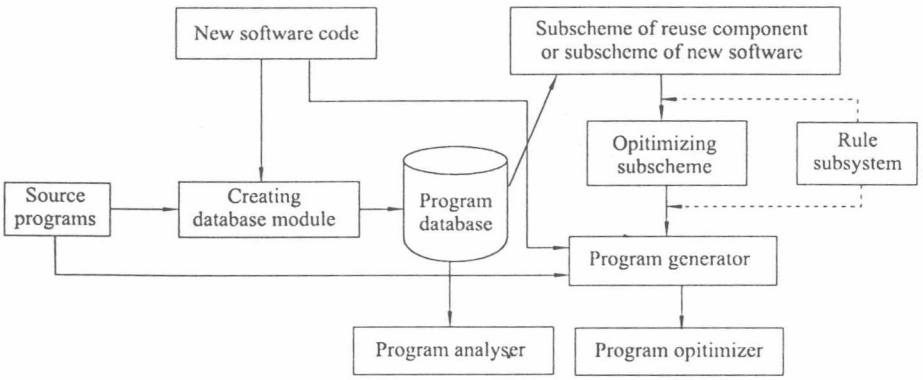

Fig. 2 The a rchitecture of CSRT system

The rectangle with dotted line is partly implemented now. For convenience, CSRT provides also the following commands:

〈Create Program DB〉 :: = CREATE DATABASE FROM [filelist | path]

〈Extract Component〉 :: = EXTRACT PROGRAM objects TO [FILE file_name | PRINT]

|EXTRACT objects

〈objects〉 :: = FILE file_name | FUNCTION function_name

| VARIABLE var_name | MACRO macro_name

| TYPE type_name

The EXTRACT command without PROGRAM only creates the relational subscheme of a reuse component.

3 PROGRAM REUSE

The program database, which is used as an interface between the CSRT and program code and between users and program code, stores the definition and reference information of all user-defined global objects. In view of database, the program database is really the conceptual scheme of entire software and a reuse component is a subscheme, in which all definition and reference information of user-defined global objects about the reuse component are stored. The program generator will extract the corresponding program code in terms of the reuse component subscheme. And then the program optimizer will generate final code of the reuse component.

3.1 Generating the component subscheme

If we want to get the component f_1, which is a function, that can be compiled and executed, we must first get a connected sub-graph, in which the vertex f_1 is its start vertex, from function call graph. This work can be transferred to solve some tuples with hierarchy information in the relation funfun. So, SELECT

statement must be extended to have the ability to deal with hierarchy information. We extend the SELECT statement to have a PARENT clause (similar to ORACLE's CONNECT clause).

The algorithm, which is used to solve the subscheme of the component f_1, is as follows:

(1) To solve all edges of the connected sub-graph with f_1 being a start vertex and store them in a relation funcall. Note that all edges of the connected sub-graph are actually some tuples in the relation funfun.

(2) All global variables, which are referenced by functions in the relation funcall, are stored in a relation fv. The INSERT statement with SELECT can do this work. Similarly, we can get all macros and types that are referred by the component f_1.

(3) Based on the global objects that are referred by the component f_1 in (2), SQL can be used to get the definition information of global objects that are needed by the f_1 on the definition relations of various objects respectively.

The definition and reference information of all global objects referred by the component f_1 consist of the subscheme of the component f_1. Users can also use a command (refer to section 3) to get the subscheme of the component f_1.

3.2 Optimizing the subscheme

If the reuse component is a set of global objects, such as a file, its subscheme may include dead code. For example, there are some global variables and functions that are only defined but not referred. In the case, we can use SQL to opitimize the subscheme. In other words, some useless tuples are deleted.

3.3 Generating program code of component

There are two approaches to reusing the program code based on the program database:

(1) he program code of the component, which can be compiled and executed, is extracted directly from the program database.

(2) Like function libraries, users can directly refer to some components, which are stored in the program database, in new C functions and then add new functions (components) in the program database. Finally, the program code of the component with function main() of the new system is generated from the program database.

Anyway, the program code must be generated from the subscheme (or scheme) of the reuse component (or new system). However, the program code of the component f_1 may have more than several thousand lines. How to extract the program code to construct a useful C file? This is concerned with many factors, such as programming style, program function, overlay, etc. Our strategies are as follows:

(1) All global variables are stored in one *H* file which is included by the C file with function main(). Other C files refer to global variables by extern statement.

(2) All global objects extracted from the program database will be stored in original files if possible. But when the number of lines in a C file is less than a threshold value, the file will be merged with the file of its brother functions in function call graph.

(3) The program optimizer is called to delete some non-reachable statements, such as the statements after statements BREAK, GOTO, CONTINUE, and RETURN or functions EXIT and ABORT.

4 RULE SUBSYSTEM

The rule subsystem is introduced in reusing a component, which is mainly used to reuse the subset of a component. Up to now, any component must be reused as a whole but users often only need to reuse the subset of a component. For example, the function A is reused but we

expect to delete the statements that call the function B in the function A. The program database can be viewed as an interface between the reuse code and original software code. Therefore, the subset of a reuse component should be represented in the subscheme of the reuse component. In other words, users can use database rules, based on the program database, to implement the subscheme of the subset of the reuse component.

Example 1 To define the rule that excludes a reference which calls function B from function A. CREATE RULE r_1

IS SELECT * FROM funcall

WHERE funname 1<>A AND funname 2 <> B

The rule r_1 will be trigged when generating the subscheme of the reuse component. Thus the tuples calling function B from function A are excluded in the subscheme of the reuse component. In generating the code of the component, the system generates the code of the subset of the reuse component in terms of the subscheme of the reuse component.

5 CONCLUSIONS

CSRT is developed on microcomputer in C. It is really an integration of our some research work, including DBMS and C software tools. CSRT is being improved on the following aspects: (1)The rule subsystem will be improved. (2)The subset reuse of the component based on the rule subsystem will be studied. This paper focuses on how to delete completely a statement or a reference from the source program. (3)The program optimizer is continuously improved. We will consider how to transfer GOTO statement to loop structure automatically or semi-automatically and how to delete useless global objects from the source program. (4)We will study and deal with C++ program.

REFERENCES

1 Arnold R S. Software reengineering. Los Altos, California: IEEE Computer Society Press, 1993

2 Arnold R S. Software reengineering: a quick history. Communications ACM. 1994, 37(5): 13～14

3 Ahrens J D. Prywes N. Lock E. Software process reengineering: toward a new generation of CASE technology. The Journal of Systems and Software. 1995, 30(1-2): 71～84

4 Wilkening D E, Loyall J P, Pitarys M J, *et al*. A reuse approach to software reengineering. The Journal of Systems and Software. 1995, 30(1-2): 117～126

5 Sage A P. Systems engineering and systems management for reengineering. The Journal of Systems and Software. 1995, 30(1-2): 3～26

6 Chen Y F, Nishimoto M Y, Ramamoorthy. The C information abstraction system. IEEE Trans on Software Engineering. 1990, 16(3): 325～334

7 Qin Xiaolin, Lin Junhai, Ye Yanfen. CMS: the C software maintenance system. Mini-Micro Systems, 1993, 14(2): 21～30

8 Laitinen K. Enhancing maintainability of source programs through disabbreviation. The Journal of Systems and Software. 1997, 37(2): 117～128

9 Maarek Y S. An information retrieval approach for automatically constructing software libraries. IEEE Trans on Software Engineering. 1991, 17(5): 800～813

第36卷 第8期 计 算 机 研 究 与 发 展 Vol. 36, NO. 8
1999年8月 JOURNAL OF COMPUTER RESEARCH & DEVELOPMENT Aug. 1999

用结构性质分开复杂类

吕义忠

(南京航空航天大学计算机科学与工程系 南京 210016)
(南京大学计算机软件新技术国家重点实验室 南京 210093)

摘 要 NP(或 Co-NP)是否包含在 P/poly 中的问题迄今仍为开问题. 80年代初证明了如果 NP⊆P/poly,则 PH $=\Sigma_2$. 最近,又有了如果 NP⊆P/poly,则 PH=ZPP 的证明. 文中将借助于广义一阶逻辑 $\mathscr{L}(\tau)$及其上的模型论以证明存在 NP(或 Co-NP)中的语言,它们没有多项式大小的线路.

关键词 结构复杂性,广义一阶逻辑,图论

中图法分类号 TP301.5

SEPARATING COMPLEXITY CLASSES BY THEIR STRUCTURAL PROPERTIES

LÜ Yi-Zhong

(*Department of Computer Science, Nanjing University of Aeronautics and Astronautics, Nanjing* 210016)
(*State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing* 210093)

Abstract Whether NP (or Co-NP)⊆P/poly is still an open problem. In the early 1980's, it has been proved that if NP⊆P/poly then PH $=\Sigma_2$. Recently, it has been proved that if NP⊆P/poly then PH=ZPP. In the paper here, it is proved by means of the extended first-order logic $\mathscr{L}(\tau)$ and model theory on it that there are some languages in NP (or Co-NP) which do not have polynomial circuits.

Key words structural complexity, extended first-order logic, graph theory

1 引 言

非一致计算模型能否判定 NP 完全语言的问题迄今尚未得到解决[1~4]. 1980年 Karp 和 Lipton[2]以及 Pippenger[5]等人引进了基于多项式长度有界函数的非一致复杂类 P/poly 并且证明它恰与多项式大小线路的复杂类相同. 与此同时,Karp,Lipton 和 Sipser 等人证明,如果 NP⊆P/poly,则 PH $=\Sigma_2$[2,6]. 当前,Köbler 和 Watanabe 证明,如果一个自可归约集有多项式大小线路,则它属于 ZPP(从而 NP),因而,PH $=\Sigma_1$[3]. 本文基于上述结果,引进了简单可数无穷图逼近接受等新概念以进一步探讨 NP(或 Co-NP)与 P/poly 之间的关系,并且借助于广义一阶逻辑 $\mathscr{L}(\tau)$及其上的模型论,证明了在 NP(或 Co-NP)中存在无多项式大小线路的语言.

原稿收到日期:1997-11-21;修改稿收到日期:1998-09-07. 本课题得到国家自然科学基金(项目编号6983^03)项目资助. 吕义忠,男,1937年生,教授,主要从事数理逻辑与计算复杂性方面的研究工作.

2 定　义

设字母表$\Sigma=\{0,1\}$,又设Σ^*为包括空字(记为"λ")在内的Σ上的一切有穷串的集合,再设N^+为全体正整数的集合,则存在双射$f:\Sigma^*\to N^+$,使得任何串$b_1\cdots b_n\in\Sigma^*$当且仅当正整数$1b_1\cdots b_n\in N^+$.因此,Σ^*中的串可按N^+排序并且特记其为$\Sigma^{+[1,4]}$.

我们定义简单可数无穷图$G=\langle V,e\rangle$,其中,顶点集V固定为可数无穷集$\{1,2,\cdots\}$,边关系e为V上的一个对称的二元关系[7,8].设$V_n=\{1,2,\cdots,n\}$,则可定义G的点诱导子图$G[V_n]=\langle V_n,e\rangle$并且简记其为$G_n$[7,8].我们称$g(n)=e(1,2)e(1,3)e(2,3)e(1,4)e(2,4)e(3,4)\cdots e(1,n)e(2,n)\cdots e(n-1,n)$为$G_n$的编码(这里视$e(i,j)$为二进值"0"或"1"并且约定$g(1)=\lambda$),又称$g$为$G$的编码函数.显然,同一个图的$e$和$g$应满足确定的关系,它可用广义一阶语言表述如下:设$G=\langle V,e\rangle$,则$g$为$G$的编码函数当且仅当$\forall_{n\in\Sigma^+}\forall_{j\leqslant n}\forall_{i<j}(BIT(\frac{1}{2}(j-1)(j-2)+i,g(n))=e(i,j))$(该事实记为(*)),这里,逻辑函数$BIT(x,y)=y$的第$x$位二进值[9,10].

我们须使用非一致复杂类P/poly,其定义为$A\in\mathrm{P/poly}\Leftrightarrow\exists\,B\in P\exists\,f\in\mathrm{poly}\forall\,x\in\Sigma^*(x\in A\Leftrightarrow\langle x,f(|x|)\rangle\in B)$,其中,poly为多项式长度有界函数集,即,$f\in\mathrm{poly}\Leftrightarrow$存在多项式$p$,使得对任何自然数$n$,$|f(n)|\leqslant p(n)$[1,4].

3 布尔线路和逼近接受

设G为简单可数无穷图(以后简称其为无穷图),又设g为G的编码函数,再设$\{C_m\}_{m\geqslant1}$为多项式大小的线路簇[1,4],则有以下定义:

定义1. 若存在$k\in N^+$使得对任何$n\in N^+$均有$\{C_m\}_{m\geqslant1}$接受$\langle k,g(n)\rangle$,则说G被$\{C_m\}_{m\geqslant1}$逼近接受.

定义2. 设C为无穷图的集合.若对任何无穷图G均有,$G\in C$当且仅当G被$\{C_m\}_{m\geqslant1}$逼近接受,则说集合C被$\{C_m\}_{m\geqslant1}$逼近接受.

引理1. 对任何集合$A\subseteq\Sigma^*$,A有多项式大小线路当且仅当$A\in\mathrm{P/poly}$[1,2,4,5].

引理2. 设C为无穷图G的集合,又设g为G的编码函数,则C被多项式大小的线路逼近接受当且仅当存在多项式时间谓词h和多项式有界函数f使得对任何无穷图G均有:$G\in C$当且仅当$\exists_{k\in\Sigma^+}\forall_{n\in\Sigma^+}h(\langle\langle k,g(n)\rangle,f(|\langle k,g(n)\rangle|)\rangle)$.

证明.(必要性)设C被多项式大小的线路簇$\{C_m\}_{m\geqslant1}$逼近接受,又设$\{C_m\}_{m\geqslant1}$接受的集合为A,则由引理1,$A\in\mathrm{P/poly}$,故按定义,存在多项式时间谓词h(P中集合B的特征函数)和多项式有界函数f,使得对任何无穷图G均有

$$\begin{aligned}G\in C&\Leftrightarrow G\text{ 被}\{C_m\}_{m\geqslant1}\text{ 逼近接受}\\&\Leftrightarrow\exists_{k\in\Sigma^+}\forall_{n\in\Sigma^+}(\langle k,g(n)\rangle\in A)\\&\Leftrightarrow\exists_{k\in\Sigma^+}\forall_{n\in\Sigma^+}h(\langle\langle k,g(n)\rangle,f(|\langle k,g(n)\rangle|)\rangle)\end{aligned}$$

(充分性)由假设条件中的h和f便能决定一个P/poly中的集合A'并且由引理1便得,A'必被某个多项式大小的线路簇$\{C'_m\}_{m\geqslant1}$所接受.因此,又由假设条件即得:$G\in C\Leftrightarrow G$被$\{C'_m\}_{m\geqslant1}$逼近接受.故依定义2,C被$\{C'_m\}_{m\geqslant1}$逼近接受.　　证毕.

4 用结构性质分开复杂类

Watanabe用"多一归约下封闭"这一结构性质分开了复杂类NP与DEXT.本文则用"不含k阶完全子图"这一结构性质来讨论NP(或Co-NP)与P/poly的分开问题.

定理1. 设$S=\{G|G$为无穷图且存在正整数r使得G不含完全子图$K_r\}$,则S不被任何多项式大小的

线路逼近接受.

证明. (反证法)反设 S 被多项式大小的线路 $\{C_m\}_{m\geq 1}$ 逼近接受,则由引理2,存在多项式谓词 h 和多项式有界函数 f,使得对任何无穷图 G 均有

$$G \in S \Leftrightarrow \exists_{k\in\Sigma^+}\forall_{n\in\Sigma^+} h(\langle\langle k,g(n)\rangle, f(|\langle k,g(n)\rangle|)) \tag{1}$$

显然,在反证法的假设下,h 和 f 是固定不变的,同样,配对函数 $\langle x,y\rangle$ 和求长度函数 $|x|$ 也是固定不变的. Immerman 在讨论有穷模型时,将固定不变的谓词和函数都归入广义一阶语言 $\mathscr{L}(\tau)$ 中(如 $BIT, s, \leqslant, o$, max 等),使它们变成 $\mathscr{L}(\tau)$ 中的逻辑谓词和逻辑函数[9]. 现在我们讨论可数模型,故同样可将固定不变的 h, f, $\langle x,y\rangle$ 和 $|x|$ 归入广义 $\mathscr{L}(\tau)$ 中. 于是可简化式(1)为

$$G \in S \Leftrightarrow \exists_{k\in\Sigma^+}\forall_{n\in\Sigma^+} A(k,g(n)) \tag{2}$$

其中,A 为广义一阶语言 $\mathscr{L}(\tau)$ 中的一阶公式.

同样地,如果将固定不变的 BIT, $D(x)$(即 $x-1$), $H(x)$(即 $x/2$)和加法"+"放入 $\mathscr{L}(\tau)$ 中,则前述(*)可简化为

$$g \text{ 为 } G \text{ 的编码函数} \Leftrightarrow \forall_{n\in\Sigma^+}\forall_{j\leqslant n}\forall_{i<j} B(i,j,g(n),e(i,j)) \tag{3}$$

其中,B 为 $\mathscr{L}(\tau)$ 上的一阶公式.

最后,显然有

$$\begin{aligned} G \text{ 含完全子图 } K_m \Leftrightarrow \exists_{n_1,\cdots,n_m\in\Sigma^+} (n_1 \neq n_2 \\ \wedge \cdots \wedge n_{m-1} \neq n_m \wedge e(n_1,n_2) \wedge \cdots \wedge e(n_{m-1},n_m)) \end{aligned} \tag{4}$$

现在,分别记式(2)、(3)和(4)右边的一阶句子为 $\psi, \chi\ \varphi_m$,则它们所含的 $\mathscr{L}(\tau)$ 中的非逻辑符号仅为 e 和 g,即型 $\tau=\{e,g\}$[9,10].

今设 $\mathscr{L}(\tau)$ 上的句子的无穷集为

$$\Gamma = \{\psi,\chi,\varphi_1,\varphi_2,\cdots,\varphi_m,\cdots\},$$

不失一般性,可设 Γ 的任一有穷子集为

$$\Gamma_m = \{\psi,\chi,\varphi_1,\varphi_2,\cdots,\varphi_m\},$$

则只要令图 G' 为由完全图 K_m 和可数无穷多个孤立点组成的无穷图,那末,由 G' 确定的 e' 和 g' 将满足 $\psi,\chi,\varphi_1,\varphi_2,\cdots,\varphi_m$. 换言之,$\Gamma$ 是有穷可满足的. 因此,由紧致性定理即得 Γ 有可数模型 u(注意:Γ 是可数的)[7]. 显然,仍可将 u 的论域视为 Σ^+. 现在可得以下3个结论:

(1) u 中的关系 e 和函数 g 应当满足 χ,故 g 为对应于 e 的无穷图的编码函数.

(2) u 应满足 ψ,故存在正整数 r,使得 e 对应的无穷图中不含 r 阶完全子图 K_r.

(3) u 应满足 $\varphi_1,\varphi_2,\cdots,\varphi_m,\cdots$,即由 e 对应的无穷图中应含 $K_1,K_2,\cdots,K_m,\cdots$. 显然,(2)和(3)矛盾,这便证明了我们的定理. 证毕.

定理2. 设 $T=\{\langle r,g\rangle | r$ 为正整数且 g 为不含子图 K_r 的无穷图的编码函数$\}$,则 $T\notin$P/poly.

证明. (反证法)反设 $T\in$P/poly,则存在多项式谓词 h 和多项式有界函数 f 使得,

$$\langle r,g\rangle \in A \Leftrightarrow h(\langle\langle r,g\rangle, f(|\langle r,g\rangle|))).$$

现在,对于定理1中的 S,我们有

$$\begin{aligned} G \in S &\Leftrightarrow \exists_{k\in\Sigma^+}(G \text{ 不含 } K_k) \\ &\Leftrightarrow \exists_{k\in\Sigma^+}\forall_{n\in\Sigma^+}(G_n \text{ 不含 } K_k) \\ &\Leftrightarrow \exists_{k\in\Sigma^+}\forall_{n\in\Sigma^+}(\langle k,g(n)\rangle \in T) \\ &\Leftrightarrow \exists_{k\in\Sigma^+}\forall_{n\in\Sigma^+}(h(\langle\langle k,g(n)\rangle\rangle, f(|\langle k,g(n)\rangle|))). \end{aligned}$$

现在,由引理2即得 S 被多项式大小的线路逼近接受,这便与定理1矛盾,得证. 证毕.

定理3. 并非 Co-NP 中的集合皆有多项式大小的线路.

证明. 已知定理2中的 $T\in$Co-NP[1,4,6]. 如果 T 有多项式大小的线路,则由引理1即得 $T\in$P/poly,这便与定理2矛盾,得证. 证毕.

定理4. 并非 NP 中的集合皆有多项式大小的线路.

证明. 因为$T\in$Co-NP,故$\bar{T}\in$NP,但$T\notin$P/poly且P/poly对补运算封闭,故$\bar{T}\notin$P/poly,即NP中集合$\bar{T}$无多项式大小的线路. 证毕.

参 考 文 献

1 Balcazar J L, Diaz J, Gabarro J. Structural Complexity. Heidelberg: Springer-Verlag, 1988(I), 1990(II)

2 Karp R, Lipton R. Some connections between non-uniform and uniform complexity classes. In: Proc of 12th ACM Symp on Theory of Computing. Heidelberg: Springer-Verlag, 1980. 302～309

3 Köbler J, Watanabe O. New collapse consequences of NP having small circuits. Lecture Notes in Computer Science #944. New York: Springer-Verlag, 1995, 196～207

4 Lü Yizhong, Sun Huicheng. The Principle of Structural Complexity. Nanjing: Nanjing University Press, 1995

5 Pippenger N. On simultaneous resource bounds. In: Proc 20th IEEE Symp Foundations of Computer Science. Heidelberg: Springer-Verlag, 1979. 307～311

6 Sipser M. A complexity theoretic approach to randomness. In: Proc 15th Ann ACM Symp on Theory of Computing. Heidelberg: Springer-Verlag, 1983. 330～335

7 Bondy J A, Murty U S R. Graph Theory with Applications. Amsterdam: North-Holland, 1976

8 Wilson R J. Introduction to Graph Theory, Third Edition, New York: Longman Inc, 1985

9 Immerman N, Descriptive and computational complexity. In: Hartmanis J ed. Computational Complexity Theory, Proc of Symposia in Applied Mathematics, Vol 38. Georgia: American Mathematical Society, 1989. 75～91

10 Malitz J. Introduction to Mathematical Logic. Heidelberg: Springer-Verlag, 1979

第 22 卷 第 4 期
1999 年 4 月

计 算 机 学 报
CHINESE J. COMPUTERS

Vol. 22 No. 4
Apr. 1999

Vakarelov 问题的研究

朱朝晖[1),2)] 毛宇光[1)] 朱梧槚[2),1)]

[1)](南京航空航天大学计算机科学研究所 南京 210016)

[2)](南京大学计算机软件国家重点实验室 南京 210093)

摘 要 本文在 Vakarelov 的信息逻辑 IL 的基础上,通过引入新的模态词[≥]以及模态词间的布尔运算建立了 BILC 系统,研究了系统的语义,提出在布尔模态逻辑语言下解决 Vakarelov 问题的思想,主要结果表明在 BILC 语言下要彻底解决 Vakarelov 问题必须针对三种 KR-结构分别构造逻辑系统,IL 系统的研究方法对 BILC 语言框架下的研究是行不通的.

关键词 知识表示系统,信息逻辑,KR-结构的扩充,Vakarelov 问题.

分类号: TP301

ON THE RESEARCH OF VAKARELOV'S OPEN PROBLEM

ZHU Zhao-Hui[1),2)] MAO Yu-Guang[1)] ZHU Wu-Jia[2),1)]

[1)](*Institute of Computer Science, Nanjing University of Aeronautics and Astronautics, Nanjing* 210016)

[2)](*State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing* 210093)

Abstract By introducing the new modality [≥] and the Boolean operators among modalities, this paper establishes information logic system BILC, and suggests that BILC is a useful tool in the researching into Vakarelov's open problem. The completeness and soundness of BILC with respect to the nonstandard frames are proved. Furthermore, the main result of the paper reveal that the method presented by Vakarelov is ill-suited for dealing with the completeness of BILC with respect to the standard frames.

Keywords Knowledge representation system, information logic, extended KR-structure, Vakarelov's open problem.

1 引 言

为了讨论知识表示的逻辑基础,Pawlak 提出了知识表示系统(Knowledge Representation system,简记为 KR-系统)及知识表示结构(Knowledge Representation structure,简记为 KR-结构)的概念. KR-系统被用来作为一些逻辑系统的语义模型[1,2],例如,DIL,NIL,INDL,IL 均以之为模型,其中,IL 是 Vakarelov 在文献[3]中提出的系统,它将 DIL,NIL,INDL 结合在一起,Vakarelov 称之为信息逻辑(Information logic). 文献[3]提出了一些开问题,其中一个问题就是在 KR-系统中引入布尔运算,对其进行逻辑刻画,即构造一个对于带布尔运算的 KR-系统是可靠且完备的逻辑系统. 本文以布尔模态逻辑语言[4]作为形式语言,对该问题做了一些研究.

本文 1998-01-20 收到,修改文 1998-11-03 收到. 本课题得到国家自然科学基金和国家八六三高技术研究发展计划资助.**朱朝晖**,男,1970 年生,获博士学位,现在南京大学做博士后研究工作,主要研究方向为人工智能理论与非经典逻辑. **毛宇光**,男,1962 年生,获博士学位,讲师,主要研究方向为数据库. **朱梧槚**,男,1935 年生,教授,博士生导师,主要研究方向为计算机科学理论与数理逻辑.

2 基本概念

2.1 知识表示系统

Pawlak 曾提出过几种知识表示系统,Orlowska 和 Pawlak 对这些系统作过较为深入的讨论,这部分工作及 KR-系统的直观介绍见文献[1,2,3],以下就本文涉及的基本概念作一简介. 如下一般性 KR-系统概念源自文献[3].

定义 1. 如果一个四元组 $S=(OB, AT, \{VAL\ a: a\in AT\}, f)$ 满足下列条件,称之为一个KR-系统:

(1) OB 是一个非空集合,其元素称为对象(Object);

(2) AT 是一个集合,其元素称为属性(Attributes);

(3) 对任意 $a\in AT$,$VAL\ a$ 是一集合,其元素称为属性 a 的值;

(4) $f: OB\times AT\to P(VAL)$,其中,$P(VAL)$ 是 VAL 的幂集,$VAL=\bigcup\{VAL\ a: a\in AT\}$;$f$ 满足如下条件:对任意 $x\in OB, a\in AT$,有 $f(x,a)\subseteq VAL\ a$.

上述定义中的函数 f 称为信息函数,根据 f 的一些性质及 VAL 的有穷性,KR-系统可分成不同类型,详见文献[3]. 下面介绍几种定义在集合 OB 上的关系:

(1) 不可区分关系 $\equiv_s: x\equiv_s y$ iff $\forall a\in AT(f(x,a)=f(y,a))$;

(2) 信息包含关系 $\leq_s: x\leq_s y$ iff $\forall a\in AT(f(x,a)\subseteq f(y,a))$;

(3) 相似关系 $\sigma_s: x\sigma_s y$ iff $\forall a\in AT(f(x,a)\cap f(y,a)\neq\varnothing)$;

(4) 确定元素集 $D_s: x\in D_s$ iff $x\in OB$ 且 $\forall a\in AT(Card(f(x,a))\leq 1)$,其中,$Card(f(x,a))$ 表示 $f(x,a)$ 的基数.

定义 2. 一个KR-系统称为可分离的当且仅当对任意 $a\in AT$ 有下列条件成立:$(\forall A,B\in VAL\ a)(\forall x\in OB)(A\in f(x,a)$ iff $B\in f(x,a)\Rightarrow A=B)$.

2.2 三种知识表示结构

定义 3. 若 $\mu=(U, \equiv, \leq, \sigma, D)$ 是一个抽象关系结构,其中 $\equiv, \sigma, \leq$ 是定义在非空集合 U 上的二元关系,D 是 U 的子集合. 若 μ 满足下列条件(s1)—(s12),则称之为一般KR-结构.

(s1) $\leq$ 具有自反性;

(s2) $\leq$ 具有可传性;

(s3) σ 具有对称性;

(s4) 若 $x\sigma y$ 则 $x\sigma x$;

(s5) 若 $x\sigma y$ 且 $x\leq z$ 则 $z\sigma y$;

(s6) 若 $y\in D$ 且 $x\leq y$ 则 $x\in D$;

(s7) 若 $x\in D$ 且 $x\sigma y$ 则 $x\leq y$;

(s8) $\equiv$ 具有自反性;

(s9) $\equiv$ 具有对称性;

(s10) $\equiv$ 具有可传性;

(s11) 若 $x\equiv y$ 则 $x\leq y$;

(s12) 若 $x\in D, y\in D$ 且 $x\sigma y$ 则 $x\equiv y$.

定义 4. 若一般KR-结构还满足下列条件(sa)(sb)则称为KR-结构.

(sa) 若 $x\leq y$ 且 $y\leq x$ 则 $x\equiv y$;

(sb) 若 $x\notin D$ 则存在 U 中元素 y 使得 $x\leq y$ 不成立.

定义 5. 设 $S=(OB, AT, \{VAL\ a: a\in AT\}, f)$ 是KR-系统,则称关系结构 $(OB, \equiv_s, \leq_s, \sigma_s, D_s)$ 为 S 上的标准KR-结构.

定理 1[3]. 任何KR-结构是一个可分离KR-系统上的标准KR-结构.

2.3 信息逻辑 IL

下面介绍 Vakarelov 的 IL 系统,有关模态逻辑的一些基本概念见文献[5,6,7]. IL 的形式语言由以下几部分组成:(1) 一个无穷的命题变元集合 VAR;(2) 逻辑联结词集 $\{\neg, \wedge, \vee\}$;(3) 模态词集 $\{[\equiv], [\leq], [\sigma]\}$;(4) 一个特别的命题常元符号 $\boldsymbol{D}$;(5) 辅助性符号(,).

IL 的合适公式集合 $FOR(\mathrm{IL})$ 定义与经典模态逻辑相类似,在此不再赘述. 为方便起见,IL 引入下列缩写符号:$true =_{\mathrm{def}} p\vee\neg p$; $false =_{\mathrm{def}} \neg true$; $A\to B =_{\mathrm{def}} \neg A\vee B$; $A\leftrightarrow B =_{\mathrm{def}} (A\to B)\wedge(B\to A)$; $\langle r\rangle A =_{\mathrm{def}} \neg[r]\neg A$, $r\in\{\equiv, \leq, \sigma\}$.

若 $\mu=(U, \equiv, \leq, \sigma, D)$ 是与KR-结构型相同的关系结构,$V: VAR\to P(U)$ 是赋值函数,则有序对 (μ, V) 是 IL 的语义模型. 对任意 $x\in U, A\in FOR(\mathrm{IL})$,按 Kripke 语义递归定义可满足关系 $x\Vdash_v A$ 如下:

(1) 若 $A\in VAR, x\Vdash_v A$ iff $x\in V(A)$;

(2) $x\Vdash_v \boldsymbol{D}$ iff $x\in D$;

(3) $x\Vdash_v \neg A$ iff $x\Vdash_v A$ 不成立;

(4) $x\Vdash_v A\wedge B$ iff $x\Vdash_v A$ 且 $x\Vdash_v B$;

(5) $x\Vdash_v [R]A$ iff $(\forall y\in U)$(若 $x R y$ 则 $y\Vdash_v A$),其中,$R\in\{\equiv, \leq, \sigma\}$.

合适公式 A 在模型 (μ, V) 中为真当且仅当 $(\forall x\in U)(x\Vdash_v A)$;$A$ 在关系结构 μ 中为真当且仅当对所有赋值函数 V,A 在模型 (μ, V) 中为真;A 在关系结构类 Σ 中为真当且仅当 A 在 Σ 的任意关系结构

μ 中为真. 下文中，将在关系结构类 Σ 中为真的所有合适公式形成的集合记作 $L(\Sigma)$，称之为 Σ 的逻辑. 定义几个关系结构类[3] 如下：Σ_0：一般 KR-结构类；Σ_1：任意 KR-系统上的标准 KR-结构类；Σ_2：任意可分离 KR-系统上的标准 KR-结构类；Σ_3：KR-结构类. 文献[3]证明了上述结构类之间具有下列关系：$\Sigma_3 = \Sigma_2 \subseteq \Sigma_1 \subseteq \Sigma_0$.

信息逻辑 IL 包含下列逻辑公理(模式)及推理规则：

(1) 经典命题逻辑的重言式；

(2) $[R](A \to B) \to ([R]A \to [R]B)$，$R \in \{\equiv, \leq, \sigma\}$；

(A_1) $[\leq]A \to A$；

(A_2) $[\leq]A \to [\leq][\leq]A$；

(A_3) $A \vee [\sigma]\neg[\sigma]A$；

(A_4) $\langle\sigma\rangle true \wedge [\sigma]A \to A$；

(A_5) $\langle\leq\rangle[\sigma]A \to [\sigma]A$；

(A_6) $\langle\leq\rangle \boldsymbol{D} \to \boldsymbol{D}$；

(A_7) $\boldsymbol{D} \wedge [\leq]A \to [\sigma]A$；

(A_8) $[\equiv]A \to A$；

(A_9) $A \vee [\equiv]\neg[\equiv]A$；

(A_{10}) $[\equiv]A \to [\equiv][\equiv]A$；

(A_{11}) $[\leq]A \to [\equiv]A$；

(A_{12}) $\boldsymbol{D} \wedge [\equiv]A \to [\sigma](\boldsymbol{D} \to A)$.

(MP) $\dfrac{A, A \to B}{B}$，

(N) $\dfrac{A}{[R]A}$，其中 $R \in \{\equiv, \leq, \sigma\}$.

记 IL 为满足下列条件的最小合适公式集：(1)IL 含所有公理；(2)IL 在(MP)，(N) 规则下封闭；(3)IL 关于代入封闭.

定理 2[3]. $IL = L(\Sigma_0) = L(\Sigma_1) = L(\Sigma_2) = L(\Sigma_3)$.

虽然 Σ_3 是 Σ_0 的真子类，但上述定理却说明在这两类关系结构中成立的永真式是相同的，导致出现这种现象的原因，我们认为在于形式语言 IL 的表达能力有限，在其中无法刻画条件(sa)，(sb). 在下文中，我们将以 ILC 为基础，通过引入模态词的布尔运算建立 BILC 语言，证明三类 KR-结构在 BILC 下得到区分，即三种 KR-结构在 BILC 下语义不等价.

3　ILC 系统

3.1　基本概念

Vakarelov 在文献[3]中提出运用布尔组合的方式扩充 KR-系统，以便刻画不同个体在属性方面的更多相互关系，并把扩充 KR-结构的公理刻画作为开问题提出来. 所谓公理刻画指构造一个模态逻辑系统 M，M 关于扩充的 KR-结构是完备与可靠的. 例如，系统 IL 就是 $\Sigma_0, \Sigma_1, \Sigma_2, \Sigma_3$ 在语言 IL 下的公理刻画. 另外，文献[3]特别指出关系‘/’所对应模态词[/]的刻画问题，其中，x/y 当且仅当 $x \leq y$ 与 $y \leq x$ 均不成立.

如果在 KR-结构中引入关系‘$\leq$’的逆关系‘$\geq$’，则‘/’的定义就变为：x/y 当且仅当 $x \leq y$ 与 $x \geq y$ 均不成立. 在布尔表达方式下即为 $/ = (-\leq) \cap (-\geq)$，其中‘$-\leq$’表示关系 $\leq$ 的补关系. 相应地，模态词[/]的刻画就成为在带布尔结构的模态逻辑中对复合模态词$[-\leq \cap -\geq]$的刻画. 这只是本文引入‘$\geq$’关系的原因之一，更深层次的原因在于，尽管关系‘$\geq$’可以用‘$\leq$’来表示，但在讨论公理刻划时模态词[$\geq$]却很难用[$\leq$]来定义，较方便的处理是把[$\geq$]作为原始模态词引入，通过公理规范[$\leq$]与[$\geq$]之间的关系. 下面，本文首先在 IL 语言中引入模态词[$\geq$]，建立 ILC 系统，并证明 ILC 关于 Vakarelov 三类 KR-结构(在其中引入关系 $\geq$)是可靠与完备的.

对任意 KR-系统 $S = (OB, AT, \{VAL\ a : a \in AT\}, f)$，在 OB 上定义一个新关系 $\geq$，它是 $\leq$ 的逆关系. 对任意关系结构 $\mu = (U, \equiv, \leq, \sigma, D)$，增添 $\leq$ 的逆关系 $\geq$ 形成的新结构，记作 μ^*，关系结构类 $\Sigma_i (0 \leq i \leq 3)$ 相应形成的新类记作 Σ_i^*.

含模态词[$\geq$]的信息逻辑系统 ILC 在语法及语义规定方面除平凡地引入关于模态词[$\geq$]的规定外，与 IL 系统没有区别. 在公理部分添加如下三条公理及一条推理规则：

(≥ 1) $A \to [\leq]\neg[\geq]\neg A$

(≥ 2) $A \to [\geq]\neg[\leq]\neg A$

(≥ 3) $[\geq](A \to B) \to ([\geq]A \to [\geq]B)$

($N \geq$) $\dfrac{A}{[\geq]A}$.

引理 1.　设 μ 是与带“$\geq$”的 KR-结构同型的关系结构，$R(\leq)$，$R(\geq)$ 分别表示在该结构中模态词[$\leq$]，[$\geq$]对应的可达关系，则

(1) 公理(≥ 1)在 μ 中为真当且仅当 $(R(\leq))^{-1} \subseteq R(\geq)$；

(2) 公理(≥ 2)在 μ 中为真当且仅当 $(R(\geq))^{-1} \subseteq R(\leq)$；

(3) 公理(≥ 1)(≥ 2)在 μ 中为真当且仅当 $(R(\geq))^{-1} = R(\leq)$，其中 $(R(\leq))^{-1}$ 表示 $(R(\leq))$

的逆关系.

证明. (3) 由(1),(2) 显然. 下面以(1) 为例进行证明. 首先证明必要性,设(≥ 1) 在 μ 中为真且 $(R(\leq))^{-1}$ 不包含于 $R(\geq)$. 从而存在 $(x,y) \in (R(\leq))^{-1}$ 且 $(x,y) \notin R(\geq)$,设 A 为原子命题符号,构造赋值函数 V 满足 $V(A) = \{z:(x,z) \notin R(\geq)\}$,所以,$x \Vdash_V [\geq]\neg A$. 由于($\geq 1$) 在 μ 中为真且 $y \Vdash_V A$,从而 $y \Vdash_V [\leq]\neg[\geq]\neg A$ 且 $x \Vdash_V \neg[\geq]\neg A$,故矛盾,因此 $(R(\leq))^{-1} \subseteq R(\geq)$. 充分性只需按 Kripke 语义进行验证即可,略. 证毕.

推论 1. 公理(≥ 1),(≥ 2) 在 Σ_i^* ($0 \leq i \leq 3$) 中均为真.

引理 2. 下列式子均为 ILC 中可证定理.

(1) $[\geq]A \rightarrow [\geq][\geq]A$

(2) $\boldsymbol{D} \rightarrow [\geq]\boldsymbol{D}$

(3) $A \rightarrow [\equiv]\langle\geq\rangle A$

(4) $[\geq]A \rightarrow A$

(5) $[\sigma]A \rightarrow [\geq][\sigma]A$

(6) $\boldsymbol{D} \wedge A \rightarrow [\sigma]\langle\geq\rangle A$

证明. 限于篇幅,略.

3.2 ILC 的典范模型

设 ILC 的极大相容集组成的类为 U_c,以 U_c 为论域构造一个关系结构 $\mu_c = (U_c, \equiv_c, \leq_c, \geq_c, \sigma_c, D_c)$ 如下:

$D_c = \{x: x \in U_c$ 且 $\boldsymbol{D} \in x\}$;

$R_c = \{\langle x,y\rangle: x,y \in U_c$ 且对任何型如 $[R]A$ 的公式,若 $[R]A \in x$ 则 $A \in y\}$,其中,$R \in \{\equiv, \leq, \geq, \sigma\}$.

根据上述构造,下面引理成立.

引理 3. (1) 在关系结构 μ_c 中,$\leq^{-1} = \geq$,即 $\leq$ 与 $\geq$ 互为逆关系.

(2) μ_c 关于语言 IL 的归约结构 $(U_c, \equiv_c, \leq_c, \sigma_c, D_c) \in \Sigma_0$.

(3) 关系结构 $\mu_c \in \Sigma_0^*$.

定义 6. 构造 μ_c 上的一个赋值函数 V_c 如下:对任意 $A \in VAR$,$V_c(A) = \{x: A \in x\}$. 称模型 (μ_c, V) 为 ILC 的典范模型.

引理 4. 在 ILC 的典范模型中,对任意 $x \in U_c$,$A \in FOR$(ILC) 有下式成立:

$$x \Vdash_{V_c} A \text{ 当且仅当 } A \in x.$$

证明. 利用极大相容集的性质及 $\sigma_c, \equiv_c, \leq_c, \geq_c, D_c$ 的构造,关于公式 A 的结构复杂度易归纳证明.

3.3 ILC 的可靠性与完备性

引理 5. 设 $\mu = (U, \equiv, \leq, \geq, \sigma, D)$ 是与带'$\geq$' 的 KR-结构同型的关系结构,则对 ILC 的公理 (A_1)—(A_{12}),A_i 在 μ 中为真当且仅当 μ 满足条件 (s_i).

证明. 由文献[3] 定理 3.1 的证明可知引理成立.

引理 6. 若 $\mu \in \Sigma_i^*$ ($i = 0,1,2,3$),则 ILC 的公理在 μ 中为真.

证明. 由推论 1,引理 5 易证.

另一方面,易于验证 ILC 推理规则的保真性,从而可得如下推论:

推论 2. ILC 关于 Σ_i^* ($i = 0,1,2,3$) 是可靠的,即 $ILC \subseteq L(\Sigma_i^*)$.

引理 7. 对任意 $A \in FOR$(ILC),若 $A \in L(\Sigma_0^*)$ 则 $A \in ILC$.

证明. 设 $A \in L(\Sigma_0^*)$ 且 $A \notin ILC$,则有一极大相容集 x 使得 $\neg A \in x$,由 μ_c 的构造及引理 4 可知在典范模型中 $x \Vdash_{V_c} \neg A$,由引理 3 可知,此与 $A \in L(\Sigma_0^*)$ 矛盾. 证毕.

定理 3. $ILC = L(\Sigma_0^*)$.

下面,我们讨论 ILC 系统相对 Σ_i^* ($i = 1, 2, 3$) 的完备性. 基本思想是由 Σ_0^* 中的模型构造与之语义等价的 Σ_3^* 中的模型. 证明的方法来自文献[3,4] 的"摹本"(Copying) 构造方法,该方法是"P-射"(P-morphism)[3,4] 技术的一般化. 设 $\mu_1 = (U_1, \equiv_1, \leq_1, \geq_1, \sigma_1, D_1)$,$\mu_2 = (U_2, \equiv_2, \leq_2, \geq_2, \sigma_2, D_2)$ 是两个关系结构,$M_1 = (\mu_1, V_1)$,$M_2 = (\mu_2, V_2)$ 分别是 μ_1, μ_2 上的模型. $I(\neq \varnothing)$ 是从 U_1 到 U_2 的函数集,对 $i \in I$,$x \in U_1$,i 作用在 x 上得到的象记作 x_i. 若对任意的 $x, y \in U_1$,$i \in I$ 有下列条件成立,则称 I 是从 μ_1 到 μ_2 的一个"摹本":

(1) $U_2 = \bigcup_{i \in I} \{x_i: x \in U_1\}$;

(2) 若 $x_i = y_j$ 则 $x = y$;

(3) 对 $R \in \{\equiv, \leq, \geq, \sigma\}$ 有:

(CR1) 若 xR_1y 则 $\exists\ j \in I(x_iR_2y_j)$;

(CR2) 若 $x_iR_2y^*$ 则 $(\exists\ j \in I)(\exists\ y \in U_1)$ $(y_j = y^*$ and $xR_1y)$;

(CD) $x \in D_1$ 当且仅当 $x_i \in D_2$.

若对 $A \in VAR$,$x \in U_1$,$i \in I$ 有 $x \in V_1(A)$ 当且仅当 $x_i \in V_2(A)$,则称 I 是从模型 M_1 到模型 M_2 的一个"摹本". 易证明下列引理成立,详细技术细节见文献[3] 引理 5.1 的证明.

引理 8. 设 $M_1 = (\mu_1, V_1)$,$M_2 = (\mu_2, V_2)$ 是两个模型,I 是从 M_1 到 M_2 的摹本,则对任意 $A \in FOR$(ILC),$x \in U_1$ 及 $i \in I$,下式成立:$x \Vdash_{V_1} A$ 当且仅当 $x_i \Vdash_{V_2} A$.