JAVA过程蓝图到抽象概念结构图的逆向映射与有效性*)

Reverse Mapping from JAVA Process Blueprin to Abstract Concept Structure Diagram and Validity

刘建宾

(汕头大学工学院计算机系　汕头515063)　(西北大学计算机系　西安710069)

Abstract The problem of consistency synchronization update to Abstract Concept Structure Diagram must be solved when edit operations to derived Abstract Logic Structure Diagram are carried out. In this paper, reverse mapping rules from logical nodes to conceptual nodes and validity theorem of bidirectional mapping functions are presented, which establish theory basis to solve the problem.

Keywords Abstract concept structure diagram, JAVA process blueprint, Abstract logic structure diagram validity, Programming

1 引言

抽象概念结构图[1]是一种不依赖于程序实现语言的概念算法描述工具,是一种用于表示程序处理过程的抽象图形化表示方法。JAVA过程蓝图[2]是与JAVA语言相关的一种程序图形表示方法,它支持JAVA程序的逻辑层和实现层表示。抽象概念结构图与JAVA过程蓝图的结合构成了JAVA程序在概念、逻辑和实现三个层次上的描述表现体系。程序设计是一个把思想变成现实、把模型变成代码的过程,是一个对设计表示进行不断修改、不断完善的动态过程。在抽象逻辑结构图[2]程序设计过程中,程序员完全可能在概念、逻辑和实现三层表示中的任何一层上进行修改,不仅可能对用抽象概念结构图表示的概念程序进行修改,而且也可能直接对已经导出的抽象逻辑结构图[1]进行修改。无论在何处进行修改,都应该保持抽象概念结构图和抽象逻辑结构图的有效性以及它们之间的一致性。这是抽象逻辑结构图程序设计中一个极其重要且必须加以解决的基本理论问题。由于逻辑层和实现层使用相同的抽象逻辑结构图进行表示,所以三层表示间的一致性实际上就是抽象概念结构图与抽象逻辑结构图之间的一致性。我们在文[1]中给出的抽象概念结构图到JAVA过程蓝图的平滑过渡方法及其概念结点到逻辑结点的映射规则,支持抽象概念结构图概念层程序表示到抽象逻辑结构图逻辑层程序表示的正向映射,使我们可以从抽象概念结构图出发导出与之一致和有效的JAVA过程蓝图,并且解决了当对抽象概念结构图进行修改时,对导出抽象逻辑结构图的一致性更新问题。但这只解决了问题的一个方面。另一方面,当程序员对导出抽象逻辑结构图进行编辑时,为了保持它与抽象逻辑结构图的一致性和有效性,必须解决对抽象概念结构图的同步更新问题。所以我们还需要在文[1]研究工作的基础上,从理论上进一步研究抽象逻辑结构图到抽象概念结构图的逆向映射问题,为抽象概念结构图的一致性同步更新方法提供理论基础。

针对上述问题,我们首先研究并提出了逻辑结点到概念结点的逆向映射规则,然后在定义概念结点与逻辑结点之间双向映射的有效性概念基础上,对双向映射有效性间的关系进行了研究,给出了双向映射的有效性定理,并通过构造与双向映射规则等价的概念结点类型与逻辑结点类型间的双向映射关系图对定理进行了证明。在逆向映射理论研究基础上,我们研究了在导出抽象逻辑结构图上执行的增、删、改三种基本编辑操作对抽象概念结构图的影响,在本文最后给出了抽象概念结构图的一致性同步更新方法。

2 逻辑结点到概念结点的逆向映射规则

设 $t_c=(A_c,dc)$ 为有效抽象概念结构图,A_c 为 t_c 的概念结点集合,$A_c\subset T_{cn}\times S_{nn}$,$T_{cn}$ 为概念结点类型的集合,S_{nn} 为表达概念结点概念语义的自然语言语句集合,dc 为 $A_c\rightarrow 2^{A_c}$ 且满足概念结点分解条件和 t_c 树条件的概念结点分解函数。我们又设 $t_l=(A_l,d_l)$ 为由 t_c 导出并经过进化或者确定化的有效并与 t_c 保持一致的抽象逻辑结构图,A_l 为 t_l 的逻辑结点集合,$A_l\subset T_{ln}\times S_l$,$T_{ln}$ 为逻辑结点类型的集合,S_l 为表达逻辑结点逻辑语义的受限自然语言集合,dl 为 $A_l\rightarrow 2^{A_l}$ 且满足逻辑结点分解条件和 t_l 树条件的逻辑结点分解函数。对于逻辑结点集合 A_l,我们把它分为导出逻辑结点集合 A_{ld} 和逻辑扩展结点集合 A_{le},$A_l=A_{ld}\cup A_{le}$。导出逻辑结点是由概念结点导出并与概念结点保持一致的逻辑结点。导出逻辑结点集合由 A_c 中的所有概念结点的导出逻辑结点组成,$A_{ld}=\{a'\mid a'\in A_l\wedge\exists\, a\ (a\in A_c)\wedge a'=f_{cl}(a)\}$。逻辑扩展结点是导出的未定义逻辑结点和未分解复合逻辑结点经确定化和分解细化而产生的逻辑子孙结点。逻辑扩展结点集合由导出抽象逻辑结构图上的所有逻辑扩展结点组成,$A_{le}=A_l-A_{ld}$。其中 f_{cl} 为 $A_c\rightarrow A_{ld}$ 的一对一且满足概念结点到逻辑结点的映射规则 R_{cl} 的映射函数。因为 f_{cl} 是 $A_c\rightarrow A_{ld}$ 的一对一映射函数,所以它存在 $A_{ld}\rightarrow A_c$ 的逆函数 f_{cl}^{-1}。

定义1 若逻辑结点 a_l 是概念结点 a_c 的导出结点,则称 a_c 为 a_l 的源概念结点。若 t_l 为抽象概念结构图 t_c 的导出抽象逻辑结构图,则称 t_c 为 t_l 的源抽象概念结构图。

下面定义逻辑结点到概念结点的逆向映射规则 $R_{cl}^{-1}=\{R_{cl}^{-1}(y)\mid y\in T_{ln}\}$ 和结点语义描述的一致性规则 R_s^{-1}。对导出逻辑结点 a',$a'\in A_{ld}$,a' 结点的概念源结点 $f_{cl}^{-1}(a')$ 应满足规则集 $R_{cl}^{-1}(a'[T_{ln}])$ 以及 R_s^{-1}。

*)国家863项目、汕头大学"211"工程项目。刘建宾　博士研究生,副教授、硕士生导师,主要从事软件方法与工具,软件工程,管理信息系统的研究工作。

·R_{cl}^{-1}(SEQ)={r1′,r2′}

r1′:|dc(f_{cl}^{-1}(a′))|≠0→f_{cl}^{-1}(a′)[T_{cn}]="+"

r2′:|dc(f_{cl}^{-1}(a′))|=0→f_{cl}^{-1}(a′)[T_{cn}]="+"∨f_{cl}^{-1}(a′)[T_{cn}]="#"

·R_{cl}^{-1}(SYN)={r3′,r4′}

r3′:|dc(f_{cl}^{-1}(a′))|≠0→f_{cl}^{-1}(a′)[T_{cn}]="&"

r4′:|dc(f_{cl}^{-1}(a′))|=0→f_{cl}^{-1}(a′)[T_{cn}]="&"∨f_{cl}^{-1}(a′)[T_{cn}]="#"

·R_{cl}^{-1}(IFT)={r5′,r6′}

r5′:|dc(f_{cl}^{-1}(a′))|≠0→f_{cl}^{-1}(a′)[T_{cn}]="o"

r6′:|dc(f_{cl}^{-1}(a′))|=0→f_{cl}^{-1}(a′)[T_{cn}]="o"∨f_{cl}^{-1}(a′)[T_{cn}]="#"

·R_{cl}^{-1}(IFE)={r7′,r8′}

r7′:|dc(f_{cl}^{-1}(a′))|≠0→f_{cl}^{-1}(a′)[T_{cn}]="?"

r8′:|dc(f_{cl}^{-1}(a′))|=0→f_{cl}^{-1}(a′)[T_{cn}]="?"∨f_{cl}^{-1}(a′)[T_{cn}]="#"

·R_{cl}^{-1}(SWH)={r9′,r10′}

r9′:|dc(f_{cl}^{-1}(a′))|≠0→f_{cl}^{-1}(a′)[T_{cn}]="?"

r10′:|dc(f_{cl}^{-1}(a′))|=0→f_{cl}^{-1}(a′)[T_{cn}]="?"∨f_{cl}^{-1}(a′)[T_{cn}]="#"

·R_{cl}^{-1}(CAS)={r11′,r12′,r13′}

r11′:brothernum(t_1,a′)=2→f_{cl}^{-1}(a′)[T_{cn}]="o"

r12′:|dc(f_{cl}^{-1}(a′))|≠0∧brothernum(t_1,a′)≥3→f_{cl}^{-1}(a′)[T_{cn}]="+"∨f_{cl}^{-1}(a′)[T_{cn}]="o"

r13′:|dc(f_{cl}^{-1}(a′))|=0∧brothernum(t_1,a′)≥3→f_{cl}^{-1}(a′)[T_{cn}]="+"∨f_{cl}^{-1}(a′)[T_{cn}]="o"∨f_{cl}^{-1}(a′)[T_{cn}]="#"

·R_{cl}^{-1}(DFT)={r14′,r15′,r16′}

r14′:brothernum(t_1,a′)=2→f_{cl}^{-1}(a′)[T_{cn}]="o"

r15′:|dc(f_{cl}^{-1}(a′))|≠0∧brothernum(t_1,a′)≥3→f_{cl}^{-1}(a′)[T_{cn}]="+"∨f_{cl}^{-1}(a′)[T_{cn}]="o"

r16′:|dc(f_{cl}^{-1}(a′))|=0∧brothernum(t_1,a′)≥3→f_{cl}^{-1}(a′)[T_{cn}]="+"∨f_{cl}^{-1}(a′)[T_{cn}]="o"∨f_{cl}^{-1}(a′)[T_{cn}]="#"

·R_{cl}^{-1}(WHL)={r17′,r18′}

r17′:|dc(f_{cl}^{-1}(a′))|≠0→f_{cl}^{-1}(a′)[T_{cn}]="*"

r18′:|dc(f_{cl}^{-1}(a′))|=0→f_{cl}^{-1}(a′)[T_{cn}]="*"∨f_{cl}^{-1}(a′)[T_{cn}]="#"

·R_{cl}^{-1}(FOR)={r19′,r20′}

r19′:|dc(f_{cl}^{-1}(a′))|≠0→f_{cl}^{-1}(a′)[T_{cn}]="*"

r20′:|dc(f_{cl}^{-1}(a′))|=0→f_{cl}^{-1}(a′)[T_{cn}]="*"∨f_{cl}^{-1}(a′)[T_{cn}]="#"

·R_{cl}^{-1}(DOW)={r21′,r22′}

r21′:|dc(f_{cl}^{-1}(a′))|≠0→f_{cl}^{-1}(a′)[T_{cn}]="*"

r22′:|dc(f_{cl}^{-1}(a′))|=0→f_{cl}^{-1}(a′)[T_{cn}]="*"∨f_{cl}^{-1}(a′)[T_{cn}]="#"

·R_{cl}^{-1}(BLK)={r23′}

r23′:f_{cl}^{-1}(a′)[T_{cn}]="!"∨f_{cl}^{-1}(a′)[T_{cn}]="#"

·R_{cl}^{-1}(DCL)={r24′}

r24′:f_{cl}^{-1}(a′)[T_{cn}]=":"∨f_{cl}^{-1}(a′)[T_{cn}]="#"

·R_{cl}^{-1}(OPE)={r25′}

r25′:f_{cl}^{-1}(a′)[T_{cn}]=":"∨f_{cl}^{-1}(a′)[T_{cn}]="#"

·R_{cl}^{-1}(LAB)={r26′}

r26′:f_{cl}^{-1}(a′)[T_{cn}]=":"∨f_{cl}^{-1}(a′)[T_{cn}]="#"

·R_{cl}^{-1}(TRY)={r27′,r28′}

r27′:|dc(f_{cl}^{-1}(a′))|≠0→f_{cl}^{-1}(a′)[T_{cn}]="e"

r28′:|dc(f_{cl}^{-1}(a′))|=0→f_{cl}^{-1}(a′)[T_{cn}]="e"∨f_{cl}^{-1}(a′)[T_{cn}]="#"

·R_{cl}^{-1}(CAH)={r29′,r30′}

r29′:|dc(f_{cl}^{-1}(a′))|≠0→f_{cl}^{-1}(a′)[T_{cn}]="o"

r30′:|dc(f_{cl}^{-1}(a′))|=0→f_{cl}^{-1}(a′)[T_{cn}]="o"∨f_{cl}^{-1}(a′)[T_{cn}]="#"

·R_{cl}^{-1}(FNY)={r31′,r32″}

r31′:|dc(f_{cl}^{-1}(a′))|≠0→f_{cl}^{-1}(a′)[T_{cn}]="+"

r32′:|dc(f_{cl}^{-1}(a′))|=0→f_{cl}^{-1}(a′)[T_{cn}]="+"∨f_{cl}^{-1}(a′)[T_{cn}]="#"

·R_{cl}^{-1}(BRK)={r33′}

r33′:f_{cl}^{-1}(a′)[T_{cn}]=">"∨f_{cl}^{-1}(a′)[T_{cn}]="#"

·R_{cl}^{-1}(CON)={r34′}

r34′:f_{cl}^{-1}(a′)[T_{cn}]=">"∨f_{cl}^{-1}(a′)[T_{cn}]="#"

·R_{cl}^{-1}(THR)={r35′}

r35′:f_{cl}^{-1}(a′)[T_{cn}]=">"∨f_{cl}^{-1}(a′)[T_{cn}]="#"

·R_{cl}^{-1}(RET)={r36′}

r36′:f_{cl}^{-1}(a′)[T_{cn}]=">"∨f_{cl}^{-1}(a′)[T_{cn}]="#"

·R_{cl}^{-1}(UND)={r37′}

r37′:f_{cl}^{-1}(a′)[T_{cn}]="#"

·结点语义描述一致性规则 R_s^{-1}={r1″}

r1″:f_{cl}^{-1}(a′)[S_{nn}]=null→f_{cl}^{-1}(a′)[S_{nn}]=a′[S_n]

3 概念结点与逻辑结点间双向映射的有效性

定义2 对 $a_c \in A_c$，若 $f_{cl}(a_c)$满足 R_{cl}，称概念结点到逻辑结点的映射函数 f_{cl}对结点 a_c 的映射是有效的。若 f_{cl}对$\forall a_c \in A_c$ 满足 R_{cl}，称概念结点到逻辑结点的映射函数 f_{cl}对概念结点集 A_c 的映射是有效的，即映射函数 f_{cl}有效。

定义3 对 $a_l \in A_{ld}$，若 $f_{cl}^{-1}(a_l)$满足 R_{cl}^{-1}，称导出逻辑结点到概念结点的逆向映射函数 f_{cl}^{-1}对导出逻辑结点 a_l 的映射是有效的。若 f_{cl}^{-1}对$\forall a_l \in A_{ld}$满足 R_{cl}^{-1}，称导出逻辑结点到概念结点的逆向映射函数 f_{cl}^{-1}对导出逻辑结点集 A_{ld}的映射是有效的，即逆向映射函数 f_{cl}^{-1}有效。

对于概念结点与逻辑结点之间双向映射函数的有效性，我们给出如下的定理：

定理1 f_{cl}有效当且仅当它的逆函数 f_{cl}^{-1}有效。

我们通过构造概念结点类型与逻辑结点类型之间的双向映射关系图 MRG(V,E)来进行证明。其中 V 是顶点的集合，E 是 V 中顶点间的有向边的集合。

首先根据概念结点到逻辑结点的映射规则 R_{cl}构造概念结点类型到逻辑结点类型的映射集合 T_{cl}，$T_{cl}=\{T_{cl}(x)|x\in T_{cn}\}=\{T_{cl}(+),T_{cl}(*),T_{cl}(?),T_{cl}(o),T_{cl}(>),T_{cl}(!),T_{cl}(:),T_{cl}(\&),T_{cl}(e),T_{cl}(\#)\}$。对概念结点类型集合 T_{cn}中的概念结点类型 x，我们可根据 $R_{cl}(x)$构造出它的逻辑结点类型映射集合 $T_{cl}(x)$，显然有 $T_{cl}(x)\subseteq T_{ln}$。具体我们有 $T_{cl}(+)=\{CAS,DFT,FNY,SEQ\}$，$T_{cl}(*)=\{WHL,FOR,DOW\}$，$T_{cl}(?)=\{IFE,SWH\}$，$T_{cl}(o)=\{CAS,DFT,CAH,IFT\}$，$T_{cl}(>)=\{BRK,CON,THR,RET\}$，$T_{cl}(!)=\{BLK\}$，$T_{cl}(:)=\{DCL,OPE,LAB\}$，$T_{cl}(\&)=\{SYN\}$，$T_{cl}(e)=\{TRY\}$，$T_{cl}(\#)=T_{ln}$。显然有所有概念结点类型的逻辑结点类型映射集合的并集$T_{cl}(+)\cup T_{cl}(*)\cup T_{cl}(?)\cup T_{cl}(o)\cup T_{cl}(>)\cup T_{cl}(!)\cup T_{cl}(:)\cup T_{cl}(\&)\cup T_{cl}(e)\cup T_{cl}(\#)=T_{ln}$。这表明规则 R_{cl}

保证 $T_{cn}\rightarrow T_{ln}$ 的映射是满上映射，同时也证明了规则 R_{cl} 对于映射 f_{cl} 的封闭性和完备性。

同理，根据逻辑结点到概念结点的逆向映射规则 R_{cl}^{-1} 构造逻辑结点类型到概念结点类型的逆向映射集合 T_{cl}^{-1}，$T_{cl}^{-1}=\{T_{cl}^{-1}(y)\mid y\in T_{ln}\}=\{T_{cl}^{-1}(SEQ),T_{cl}^{-1}(SYN),T_{cl}^{-1}(IFT),T_{cl}^{-1}(IFE),T_{cl}^{-1}(SWH),T_{cl}^{-1}(CAS),T_{cl}^{-1}(DFT),T_{cl}^{-1}(WHL),T_{cl}^{-1}(FOR),T_{cl}^{-1}(DOW),T_{cl}^{-1}(BLK),T_{cl}^{-1}(DCL),T_{cl}^{-1}(OPE),T_{cl}^{-1}(LAB),T_{cl}^{-1}(TRY),T_{cl}^{-1}(CAH),T_{cl}^{-1}(FNY),T_{cl}^{-1}(BRK),T_{cl}^{-1}(CON),T_{cl}^{-1}(THR),T_{cl}^{-1}(RET),T_{cl}^{-1}(UND)\}$。对逻辑结点类型集合 T_{ln} 中的逻辑结点类型 y，我们可根据规则集 $R_{cl}^{-1}(y)$ 构造出它的概念结点类型逆向映射集合 $T_{cl}^{-1}(y)$，显然有 $T_{cl}^{-1}(y)\subseteq T_{cn}$。具体我们有 $T_{cl}^{-1}(SEQ)=\{+,\#\}$，$T_{cl}^{-1}(SYN)=\{\&,\#\}$，$T_{cl}^{-1}(IFT)=\{o,\#\}$，$T_{cl}^{-1}(IFE)=\{?,\#\}$，$T_{cl}^{-1}(SWH)=\{?,\#\}$，$T_{cl}^{-1}(CAS)=\{o,+,\#\}$，$T_{cl}^{-1}(DFT)=\{o,+,\#\}$，$T_{cl}^{-1}(WHL)=\{*,\#\}$，$T_{cl}^{-1}(FOR)=\{*,\#\}$，$T_{cl}^{-1}(DOW)=\{*,\#\}$，$T_{cl}^{-1}(BLK)=\{!,\#\}$，$T_{cl}^{-1}(DCL)=\{:,\#\}$，$T_{cl}^{-1}(OPE)=\{:,\#\}$，$T_{cl}^{-1}(LAB)=\{:,\#\}$，$T_{cl}^{-1}(TRY)=\{e,\#\}$，$T_{cl}^{-1}(CAH)=\{o,\#\}$，$T_{cl}^{-1}(FNY)=\{+,\#\}$，$T_{cl}^{-1}(BRK)=\{>,\#\}$，$T_{cl}^{-1}(CON)=\{>,\#\}$，$T_{cl}^{-1}(THR)=\{>,\#\}$，$T_{cl}^{-1}(RET)=\{>,\#\}$，$T_{cl}^{-1}(UND)=\{\#\}$。容易验证所有逻辑结点类型的概念结点类型映射集合的并集 $T_{cl}^{-1}(SEQ)\cup T_{cl}^{-1}(SYN)\cup T_{cl}^{-1}(IFT)\cup T_{cl}^{-1}(IFE)\cup T_{cl}^{-1}(SWH)\cup T_{cl}^{-1}(CAS)\cup T_{cl}^{-1}(DFT)\cup T_{cl}^{-1}(WHL)\cup T_{cl}^{-1}(FOR)\cup T_{cl}^{-1}(DOW)\cup T_{cl}^{-1}(BLK)\cup T_{cl}^{-1}(DCL)\cup T_{cl}^{-1}(OPE)\cup T_{cl}^{-1}(LAB)\cup T_{cl}^{-1}(TRY)\cup T_{cl}^{-1}(CAH)\cup T_{cl}^{-1}(FNY)\cup T_{cl}^{-1}(BRK)\cup T_{cl}^{-1}(CON)\cup T_{cl}^{-1}(THR)\cup T_{cl}^{-1}(RET)\cup T_{cl}^{-1}(UND)=T_{cn}$。这表明规则 R_{cl}^{-1} 保证 $T_{ln}\rightarrow T_{cn}$ 的映射是满上映射，同时也证明了规则 R_{cl}^{-1} 对于逆向映射 f_{cl}^{-1} 的封闭性和完备性。

令 $V_c=T_{cn}$，$V_d=T_{ln}$，$V=V_c\cup V_d=T_{cn}\cup T_{ln}$，得到有向映射关系图 MRG 的顶点集合 V。V 是概念结点类型顶点集合 V_c 和逻辑结点类型顶点集合 V_d 的并集。

下面我们构造映射关系图 MRG 的有向边集合 E。E 由概念结点类型顶点到逻辑结点类型顶点的有向边集合 E_{cl} 和逻辑结点类型顶点到概念结点类型顶点的有向边集合 E_{lc} 组成，$E=E_{cl}\cup E_{lc}$。E_{cl} 是由 V_c 中顶点 $x\in V_c$ 和 V_d 中顶点 $y\in V_d$ 所组成有效顶点对 (x,y) 的集合，它是 $V_c\times V_d$ 的子集，$E_{cl}\subset V_c\times V_d$。对 V_c 中的概念结点类型顶点 x，我们可以构造从顶点 x 出发到 V_d 中逻辑结点类型顶点的有向边集 $E_{cl}(x)$。$E_{cl}(x)$ 为概念结点类型顶点 x 同它的逻辑结点类型映射集合 $T_{cl}(x)$ 中的各个逻辑结点类型顶点配对形成的顶点对集，$E_{cl}(x)=\{(x,y)\mid y\in T_{cl}(x)\}$。将 V_c 中各个概念顶点出发的有向边集构造出来就可得到 E_{cl}。E_{cl} 是所有概念顶点出发的有向边集的并集，$E_{cl}=E_{cl}(+)\cup E_{cl}(*)\cup E_{cl}(?)\cup E_{cl}(o)\cup E_{cl}(>)\cup E_{cl}(!)\cup E_{cl}(:)\cup E_{cl}(\&)\cup E_{cl}(e)\cup E_{cl}(\#)=\{(x,y)\mid x\in V_c\wedge y\in T_{cl}(x)\}=\{(x,y)\mid x\in T_{cn}\wedge y\in T_{cl}(x)\}$。同理，我们可根据逻辑结点类型顶点集合 V_d 和逻辑结点类型到概念结点类型的逆向映射集合 T_{cl}^{-1} 构造出逻辑结点类型顶点到概念结点类型顶点的有向边集合 E_{lc}，E_{lc} 是由 V_d 中顶点 $y\in V_d$ 和 V_c 中顶点 $x\in V_c$ 所组成有效顶点对 (y,x) 的集合，它是 $V_d\times V_c$ 的子集，$E_{lc}\subset V_d\times V_c$。对 V_d 中的逻辑结点类型顶点 y，我们可以构造从顶点 y 出发到 V_c 中概念结点类型顶点的有向边集 $E_{lc}(y)$。$E_{lc}(y)$ 为逻辑结点类型顶点 y 同它的概念结点类型逆向映射集合 $T_{cl}^{-1}(y)$ 中的各个概念结点类型顶点配对形成的顶点对集，$E_{lc}(y)=\{(y,x)\mid x\in T_{cl}^{-1}(y)\}$。将 V_d 中各个逻辑顶点出发的有向边集构造出来就得到 E_{lc}，E_{lc} 是所有逻辑顶点出发的有向边集的并集，$E_{lc}=E_{lc}(SEQ)\cup E_{lc}(SYN)\cup E_{lc}(IFT)\cup E_{lc}(IFE)\cup E_{lc}(SWH)\cup E_{lc}(CAS)\cup E_{lc}(DFT)\cup E_{lc}(WHL)\cup E_{lc}(FOR)\cup E_{lc}(DOW)\cup E_{lc}(BLK)\cup E_{lc}(DCL)\cup E_{lc}(OPE)\cup E_{lc}(LAB)\cup E_{lc}(TRY)\cup E_{lc}(CAH)\cup E_{lc}(FNY)\cup E_{lc}(BRK)\cup E_{lc}(CON)\cup E_{lc}(THR)\cup E_{lc}(RET)\cup E_{lc}(UND)=\{(y,x)\mid y\in V_d\wedge x\in T_{cl}^{-1}(y)\}=\{(y,x)\mid y\in T_{ln}\wedge x\in T_{cl}^{-1}(y)\}$。

上面我们已经构造出 MEG 的顶点集合 V 和有向边集合 E，现可按下面的方法画出有向图 MRG。首先画出 V 中的所有顶点，因为 $V=V_c\cup V_d$，所以可将 V 的顶点画成两排，一排为 V_c，另一排为 V_d。第二步画出 V_c 到 V_d 的有向边集 E_{cl}。对 $x\in V_c$，$y\in V_d$，若顶点对 $(x,y)\in E_{cl}$，定有 $(x,y)\in E_{cl}(x)$，画一条顶点 x 到顶点 y 的有向边，并在这条有向边上标注出规则集 $R_{cl}(x)$ 中相应规则的映射条件。第三步可按与第二步相同的方法画出 V_d 到 V_c 的有向边集 E_{lc}。对 $y\in V_d$，$x\in V_c$，若顶点对 $(y,x)\in E_{lc}$，定有 $(y,x)\in E_{lc}(y)$，画一条顶点 y 到顶点 x 的有向边，并在这条有向边上标注出规则集 $R_{cl}^{-1}(y)$ 中相应规则的映射条件。

从上面 MRG(V,E) 的构造和绘制方法，可以看到：顶点集 V_c 表示了所有的概念结点类型 T_{cn}，顶点集 V_d 表示了所有的逻辑结点类型 T_{ln}；V_c 到 V_d 的有向边表示了概念结点类型到逻辑结点类型的映射关系及其条件，而 V_d 到 V_c 的有向边表示了逻辑结点类型到概念结点类型的映射关系及其条件；带条件的有向边集 E_{cl} 表示了规则 R_{cl} 中所有规则体现的映射关系及映射条件，而带条件的有向边集 E_{lc} 表示了规则 R_{cl}^{-1} 中所有规则体现的逆向映射关系及其条件。所以按上面构造和绘制出来的带条件的有向映射关系图 MRG(V,E) 是规则 R_{cl} 和 R_{cl}^{-1} 的等价表示。

带条件的有向关系映射图 MRG($V=V_c\cup V_d$，$E=E_{cl}\cup E_{lc}$) 具有如下特性：(1)对 V 中的任何概念结点类型顶点 x 和任何逻辑结点类型顶点 y，即 $x\in V_c$，$y\in V_d$，若 $(x,y)\in E_{cl}$ 是 E 中的有向边，则一定存在 $(y,x)\in E_{lc}$ 是 E 中的有向边。反之，对 V 中的任何逻辑结点类型顶点 y 和任何概念结点类型顶点 x，$y\in V_d$，$x\in V_c$，若 $(y,x)\in E_{lc}$ 是 E 中的有向边，则一定存在 $(x,y)\in E_{cl}$ 是 E 中的有向边。这表明对于 V 中的任意两个概念结点类型顶点和逻辑结点类型顶点，若它们之间存在映射关系，则它们的映射关系一定是双向的。即 E 中由顶点对表示的所有有向边具有对称性。在图上可清楚地看到所有存在映射关系的两个顶点间均有两条方向相反的有向边构成一个环。(2)如果 (x,y) 和 (y,x) 是顶点 $x\in V_c$ 和顶点 $y\in V_d$ 间的一对方向有向边，则有向边 (x,y) 上的映射条件与有向边 (y,x) 上的映射条件相互相容。即若有向边 (x,y) 上的映射条件成立，则有向边 (y,x) 上的映射条件也成立；反之，若有向边 (y,x) 上的映射条件为真，则有向边 (x,y) 上的映射条件也为真。

利用映射关系图 MRG($V=V_c\cup V_d$，$E=E_{cl}\cup E_{lc}$) 及其特性，容易证明定理成立。首先证明若 f_{cl} 有效则它的逆函数 f_{cl}^{-1} 有效，若 f_{cl} 有效，则 f_{cl} 对 A_c 中的任一结点有效。设 a 是 A_c 中的任一结点，因 f_{cl} 有效，根据定义有 f_{cl} 对于 a 有效，即 $f_{cl}(a)$ 满足 R_{cl}。这意味着可从关系图 MRG 的一个概念顶点 $a[T_{cn}]$ 出发经过有向边 $(a[T_{cn}],f_{cl}(a)[T_{ln}])$ 而达到关系图 MRG 的另一逻辑顶点 $f_{cl}(a)[T_{ln}]$，并使有向边 $(a[T_{cn}],f_{cl}(a)[T_{ln}])$ 上的映射条件为真。由关系图 MRG 的特性知：(1)必然存在逻辑

顶点 $f_{cl}(a)[T_{ln}]$到概念顶点 $a[T_{cn}]$的另一有向边($f_{cl}(a)[T_{ln}]$, $a[T_{cn}]$)使这两个顶点构成环。由关系图 MRG 的特性(2)知有向边($f_{cl}(a)[T_{ln}]$,$a[T_{cn}]$)上的映射条件也为真。因 $a[T_{cn}]=f_{cl}^{-1}(f_{cl}(a))[T_{cn}]$,换句话说就是 $f_{cl}^{-1}(f_{cl}(a))$满足 R_{cl}^{-1},即 f_{cl}的逆向映射函数 f_{cl}^{-1}对于 a 的导出逻辑结点 $f_{cl}(a)$有效,从而证明了若 f_{cl}有效则它的逆函数 f_{cl}^{-1}有效。同理可证,若 f_{cl}^{-1}有效则 f_{cl}有效。定理成立。

该定理给出了概念结点与逻辑结点之间双向映射函数有效性的充分必要条件并保证了规则集 R_{cl}和 R_{cl}^{-1}之间的无矛盾性。

4 抽象概念结构图的一致性同步更新方法

当程序员对抽象逻辑结构图进行编辑时,为了保持抽象概念结构图与抽象逻辑结构图的一致性,必须考虑对抽象概念结构图的同步更新问题。对抽象逻辑结构图的基本编辑操作包括增加逻辑结点、删除逻辑结点、以及修改逻辑结点。每种基本操作的执行都会使逻辑程序表示从一个状态改变到另一个新的状态。假定在此之前,抽象逻辑结构图与抽象概念结构图两者保持一致,由于某种编辑操作已使抽象逻辑结构图发生改变,此时的抽象概念结构图可能变得"过时"而与抽象逻辑结构图的新状态不相一致。因而必须对"过时"的抽象概念结构图进行相应的更新,从不一致的状态调整更新到相一致的状态。

下面我们分情况来讨论每种编辑操作对抽象概念结构图的影响,并给出对抽象概念结构图的一致性同步更新方法。

4.1 逻辑结点的添加

设 a' 是要添加的逻辑结点,$p'=parent(t_l,a')$是 a' 结点的父结点,我们分下面三种情况来讨论:

(1)如果 p' 是导出逻辑结点,即存在一概念结点 $p,p\in A_c$,有 $p'=parent(t_l,a')=f_{cl}(p)$,$p=f_{cl}^{-1}(p')$,且 p' 的概念源结点 p 是已分解复合结点,即 $dc(p)\neq\varnothing$,$|dc(p)|=|dc(f_{cl}^{-1}(p'))|\neq 0$,即 p 存在至少一个以上的概念孩子结点,由导出方法知这些概念孩子结点导出的逻辑结点就是要添加的逻辑结点 a 的兄弟结点,换句话说就是要添加的逻辑结点 a 的兄弟结点均为导出结点。因导出结点的兄弟结点均为导出结点,所以要添加的逻辑结点 a 本身也将为导出结点。这种情况下添加逻辑结点 a 必然导致新的抽象逻辑结构图 t_l' 与抽象概念结构图 t_c 不一致。对抽象概念结构图 t_c 必须进行更新,添加 p 结点的一个孩子结点即添加 a' 逻辑结点的抽象概念结点。具体方法是创建一个新的概念结点 a,$|dc(a)|=0$,使 a 成为 p 结点的孩子结点,更新 $dc(p)=dc(p)\cup\{a\}$,并建立 a' 与 a 结点之间的双向映射关系,即使 $a=f_{cl}^{-1}(a')$,$a'=f_{cl}(a)$,并使 $f_{le}^{-1}(a')$满足上面给出的逻辑结点到概念结点的逆向映射规则集 R_{cl}^{-1}和语义规则 R_s^{-1},以确定概念结点 a 的有效概念类型和语义。这样就可得到与抽象逻辑结构图 t_l' 保持一致的抽象概念结构图 t_c'。对 t_c 的更新完成添加的逻辑结点 a' 就成为新增抽象概念结点 a 的导出结点,有 $a'\in A_{le}$。

(2)如果要添加的逻辑结点 a' 的父结点 p' 是逻辑导出结点,且 p' 的概念源结点 p 是未分解复合结点或者是未定义结点,即 $dc(p)=\varnothing$,$|dc(p)|=|dc(f_{cl}^{-1}(p'))|=0$,即 p 无任何概念孩子结点,由导出方法知 p' 的所有子孙结点均为扩展结点,要添加的逻辑结点 a' 也为扩展结点,即执行逻辑结点添加操作后有 $a'\in A_{le}$。对抽象逻辑结构图 t_l 添加逻辑扩展结点 a' 不会对抽象概念结构图 t_c 产生影响,得到的新抽象逻辑结构图 t_l' 仍然与 t_c 保持一致。我们可令 $t_c'=t_c$,就有 t_c' 与 t_l' 相一致,即我们不需对 t_c 进行任何更新操作而保持原来 t_c 的状态就可得到与抽象逻辑结构图 t_l' 保持一致的新抽象概念结构图 t_c'。

(3)如果 p' 是逻辑扩展结点,即 t_c 的概念结点集合 Ac 中不存在任何概念结点 p,使 $p'=parent(t_l,a')=f_{cl}(p)$。扩展结点的子孙结点均为扩展结点,a' 是 p' 的孩子结点,所以要添加的逻辑结点 a' 是扩展结点,即执行逻辑结点添加操作后有 $a\in A_{le}$。这种情况同上面(2)讨论的情况一样,对抽象逻辑结构图 t_l 添加逻辑扩展结点 a' 不会对与抽象概念结构图 t_c 的一致性产生影响,从而并不需要执行对抽象概念结构图 t_c 的任何更新操作。

4.2 逻辑结点的删除

对一个树结点的删除,如果被删除的结点无孩子结点,则只需删除结点本身,如果被删除的结点是带有孩子结点的复合结点,则会引起连锁反应。因这时要删除的是以删除结点为根的子树,即不仅要删除结点本身,还要删除结点的所有子孙结点。

假设 a' 是抽象逻辑结构图 t_l 上要被删除的结点,下面我们依 a' 是导出结点还是扩展结点而分成两种情况来讨论:

(1)如果 a' 是逻辑导出结点,即 $a'\in A_{ld}$,则在抽象概念结构图 t_c 的概念结点集 Ac 中必存在一个概念结点 a,有 $a'=f_{cl}(a)$。删除逻辑层的导出结点 a',必然导致删除后的抽象逻辑结构图 t_l' 与抽象概念结构图 t_c 不一致,必须将导出结点 a' 的概念源结点 a 从 t_c 中删除。若 a' 为无孩子的结点,即 a' 是逻辑叶结点或未分解逻辑复合结点,$|dl(a')|=0$,由假设 t_c 与 t_l 相一致知 a' 的概念源结点 a 为概念叶结点或未分解概念复合结点,也为无孩子的结点即 $|dc(a)|=0$,所以只需对抽象概念结构图 t_c 执行结点 a 的删除操作 $A_c=A_c-\{a\}$,并对 a 结点的父结点 parent(t_c,a)的结点分解函数执行更新操作 dc(parent(t_c,a))=dc(parent(t_c,a))$-\{a\}$,将 t_c 中的概念结点 a 删除。若 a' 为有小孩结点的已分解复合逻辑结点,$|dl(a')|\neq 0$,则除了删除 a' 结点之外,还要删除 a' 结点的所有子孙结点。由假设 t_c 与 t_l 一致知 a' 的概念源结点 a 为已分解概念复合结点,有 $|dc(a)|\neq 0$,对抽象概念结构图 t_c 的更新除了要删除 a' 的概念源结点 a 外,同样需要将 a 的所有子孙结点删除。为此需要构造要删除的概念结点集合 $D_c(a)$,初始时将 a 加入 $D_c(a)$,使 $D_c(a)=\{a\}$,然后用遍历算法遍历以 a 为根的子树,将 a 的所有子孙结点加入 $D_c(a)$中。构造好 $D_c(a)$后,将 $D_c(a)$中的每个结点从 t_c 中删除,t_c 的概念结点集合 A_c 更新为 $A_c=A_c-D_c(a)$,然后再更新 a 结点的父结点 parent(t_c,a)的结点分解函数使 dc(parent(t_c,a))=dc(parent(t_c,a))$-\{a\}$。用这样的方法更新 t_c 后就可得到与 t_l' 保持一致的新的抽象概念结构图 t_c'。

(2)如果 a' 是逻辑扩展结点,即 t_c 的概念结点集合 A_c 中不存在任何概念结点 a,使 $a'=f_{cl}(a)$。因扩展结点的子孙结点均为扩展结点,所以无论 a' 是否有孩子结点,要删除的结点均为逻辑扩展结点。删除逻辑扩展结点不会对抽象逻辑结构图与抽象概念结构图的一致性产生任何影响。这种情况下,对 a' 结点及其子孙结点的删除不会引起对抽象概念结构图 t_c 的任何一致性更新操作。

4.3 逻辑结点的修改

逻辑结点的修改可以是概念结点类型的修改和结点逻辑

(下转第18页)

间提供一个中间件，使得使用不同硬件环境、操作系统环境的各个用户能够透明而安全地提交自己的计算请求；由中间件将这些计算请求转换成超级计算机上预登记应用程序所要求的形式，交给超级计算机进行计算。

在上述网格计算模型中，基本上没有针对网格计算内核做什么工作，随着 Jini[7]技术的出现，前端技术的研究变得更为简单，而内核技术的研究则相对落后。高性能计算由局域网范围扩展到广域网时缺乏适合广域网计算并行程序编程模型和统一的、与编程模型相协调的计算资源管理模型。

针对网格计算系统的特点及目前研究的实际情况，一些研究单位将网络计算的前端与内核统一看待，研究网格计算系统资源的异构与动态特性、计算资源与并行任务充分匹配的资源管理模型，提出并研究一种不依赖用户身份的网格计算系统，并且给出安全、可靠、低开销的可定制资源保护策略，使网格计算系统资源管理与资源保护更加实用、科学。

4. 网格计算系统的研究趋势

分析网格计算系统的特点及目前研究的实际情况，网格技术未来的研究可能集中在如下几个方面：

(1)*动态自适应性研究*　研究网格计算系统的动态自适应性，使得网格计算系统能够自动适应环境的变化。网格计算系统中某一资源出现故障或失败的可能性较高，系统的资源会不断扩大、应用会不断增长，系统的整体结构和整体性能会不断地发生变化，并且随时有不可预测的系统行为发生，这就要求资源管理程序能够动态监视和管理网格资源，从目前可利用的资源中选取最佳资源服务，尽量减小由于这种故障或失败、整体结构和整体性能发生变化或不可预测的系统行为等问题对网格整体性能的影响。

(2)*安全管理研究*　研究网格计算系统的安全管理机制，确保网格计算系统管理和使用的安全性。建立全网格的帐号管理和记账系统，使得任何用户能够从任何连在网格的计算机上，安全登录并有效使用网格资源；确定适用于网格计算系统的信息加密机制和信息传输机制；确定网格计算系统的管理层次体系，将管理域按照区域层次划分，并且决定管理信息流的流向；能够为不同级别的系统管理员提供强有力的工具或界面监视系统资源和系统的运行情况。

(3)*应用计算研究*　研究可使用网格计算系统进行计算的新应用，充分利用网络上的各种资源来支持大型的并行分布式计算，由应用驱动来提出对网格计算系统的功能和技术要求，并验证其技术途径和技术实现的有效性。

(4)*高效的程序编译模型和执行引擎研究*　研究解决平台无关的中间代码执行效率低下的问题，对程序的编译模型和执行引擎进行研究，解决平台无关性和执行效率之间的矛盾，最终能够提供一种或几种解决方案，使得程序能够顺利地在异构环境下执行和使用系统资源。

参 考 文 献

1 Smarr J, Catlett C. Metacomputing. Communication of the ACM, 1992, 35: 44～52

2 Foster I, Kesselman C. Globus: A Metacomputing Infrastructure Toolkit. International Journal of Supercomputer Applications, 1997, 11(2): 115～128

3 http:// www. origincom. com. cn/article/20010627-1. htm

4 http://www. ccw. com. cn/htm/produ/topic/01_6_8_13. asp

5 http://www. dl. ac. uk/TCSC/UKHEC/GridWorkshop/Tanaka/ninf/img2. htm

6 http://www. grid. org. cn/html/overview/info. htm

7 http://www. jini. org/whatisjini. html

(上接第146页)

语义的修改，这两种修改操作都可改变抽象逻辑结构图 t_l 的状态，但不会改变其结构。若逻辑结点的修改操作引起概念结点类型和概念结点语义的变化，则会导致抽象概念结构图 t_c 的状态变化。由于逻辑结点的修改不会引起抽象逻辑结构图 t_l 的结构改变，因而逻辑结点的修改不会导致抽象概念的结构变化。

假设 a' 是抽象逻辑结构图 t_l 上将被修改的逻辑结点，下面我们依据 a' 是导出结点还是扩展结点而分成两种情况来讨论：

(1)如果 a' 是逻辑导出结点，即 $a' \in A_{ld}$，则在抽象概念结构图 t_c 的概念结点集 A_c 中存在一个概念结点 a，有 $a' = f_{cl}(a)$，$a = f_{cl}^{-1}(a')$。修改逻辑导出结点 a' 的逻辑结点类型，必然引起抽象逻辑结构图 t_l 的状态变化而导致修改后的抽象逻辑结构图 t_l' 与抽象概念结构图 t_c 中结点 a 的概念结点类型与 a' 的逻辑结点类型不一致，必须对导出结点 a' 的概念源结点 a 的概念结点类型进行更新。这时只需使 $a = f_{cl}^{-1}(a')$ 满足逻辑结点到概念结点的逆向映射规则 R_{cl}^{-1}，以确定 a 结点的逻辑结点类型 a[Tcn]。若对结点 a' 的修改有效而保证修改后的抽象逻辑结构图 t_l' 的有效性，我们前面给出的定理就能够保证对 a' 的概念源结点 a 的上述更新方法的有效性，从而使新的抽象概念结构图 t_c' 与新的抽象逻辑结构图 t_l' 保持一致。对逻辑导出结点 a' 的逻辑结点语义的修改以及由此而引起的概念结点 a' 的概念语义的一致性更新属于语义范畴，主要由人工进行维护。

(2)如果 a' 是逻辑扩展结点，即 t_c 的概念结点集合 Ac 中不存在任何概念结点 a，使 $a' = f_{cl}(a)$。因修改逻辑扩展结点不会对抽象逻辑结构图与抽象概念结构图的一致性产生任何影响，所以对 a' 结点的逻辑结点类型和逻辑结点语义的修改均不会引起对抽象概念结构图 t_c 的任何一致性更新操作。

结束语　本文提出的逻辑结点到概念结点的逆向映射规则以及双向映射函数的有效性定理为抽象逻辑结构图编辑操作引起的抽象概念结构图一致性更新问题的解决提供了理论基础，是抽象逻辑结构图可视化程序设计方法学中的重要内容，对于维护抽象概念结构图与导出抽象逻辑结构图的有效性与两者间的一致性具有重要意义。文中证明的双向映射函数的有效性的定理保证了两个规则集的无矛盾性和映射的封闭性。双向映射体系的构建，使程序员在设计过程中既可对抽象概念结构图进行编辑，又可对导出抽象逻辑结构图进行修改，使抽象概念结构图和抽象逻辑结构图对程序设计过程的支持变得非常灵活。

参 考 文 献

1 刘建宾，郝克刚，龚世生. 抽象概念结构图到 JAVA 过程蓝图的平滑过渡及一致性. 计算机科学，2001，28(8)

2 刘建宾. JAVA 过程蓝图. 计算机科学，2000，27(7)