# HOS 规格说明的功能理解及其应用*

吕 建　　费宗铭

(南京大学计算机软件研究所,南京 210093)

**摘要**　本文给出了 HOS 方法学中基本控制结构 JOIN、INCLUDE、OR 和复合控制结构 COJOIN、COINCLUDE 及 COOR 的语义构造规则. 以此为基础,提出了一种层次式功能理解方法,并讨论了其在 HOS 规格说明的语义验证和复用方面的应用.

**关键词**　HOS 规格说明,功能理解,验证,复用.

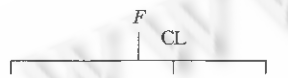


图1

HOS 方法学是详细规格说明和开发软件系统的十分有效的方法[1,2]. 其基本思想是以函数作为软件系统的数学模型,软件设计由一系列自顶向下的精化步组成,每一精化步可表示如图 1. (其中 $F$ 和 $F_i(1\leqslant i\leqslant n)$表示函数,$CL$ 表示控制结构). 它将给定的函数分解成若干子函数. 这种分解过程一直进行到所有子函数的功能已知为止. 结果,形成一棵描述整个软件系统的树形结构—HOS 软件规格说明.

为了保证软件系统的可靠性,HOS 方法学基于 6 条公理严格定义了控制结构 CL. CL 主要包括 3 个基本控制结构 JOIN、INCLUDE 和 OR 以及定义在基本控制结构之上的复合控制结构 COJOIN、COINCLUDE 和 COOR,使得每一精化步可完成函数的二叉分解并保证所开发出的 HOS 规格说明的接口清晰且正确.

然而,由于缺乏显式的、形式的函数功能规格说明机制和控制结构的严格的语义描述,HOS 方法仅能保证每一分解步的接口正确性而不能保证其语义正确性.

为此,我们引入了基于一阶谓词演算的前后断言方法作为函数的功能规格说明机制,并基于前件与后件推导的概念,分别给出了 HOS 方法学中各种控制结构的语义构造规则,提出了一种能在某种意义下验证 HOS 规格说明语义正确性的层次式功能理解方法. 在此基础上,本文还讨论了 HOS 规格说明自动复用的问题.

## 1　基本概念

**定义** 1.1. 函数 $F$ 的说明是一个三元组,即 $FD(F)=\langle FM(F),DO_F,RA_F\rangle$,其中 $FM(F)$表示函数格式,其一般形式是 $Y=F(X)$;$F$ 是函数名,$X$ 是输入变量元组,$Y$ 是输出变量

* 本文 1994－08－13 收到,1994－11－18 定稿

作者吕建,1960 年生,教授,主要研究领域为软件自动化. 费宗铭,1965 年生,博士生,主要研究领域为软件自动化.

本文通讯联系人:吕建,南京 210093,南京大学计算机软件研究所

元组,即 $X=(x_1,\cdots,x_m)$,$Y=(y_1,\cdots,y_n)$,$(m,n\geqslant 1)$;$DO_F$ 表示函数 $F$ 的输入的取值范围,其一般形式是 $D_1\times\cdots\times D_m$;$RA_F$ 表示函数 $F$ 的输出的取值范围,其一般形式是 $R_1\times\cdots\times R_n$;每一 $D_i(1\leqslant i\leqslant m)$与 $R_j(1\leqslant j\leqslant n)$均由数据类型刻划.

**定义** 1.2. 函数 $F$ 的功能规格说明是一个二元组,即 $P(F)=\langle FD(F),SP_F\rangle$,其中 $FD(F)$是函数 $F$ 的说明,$SP_F$ 是函数 $F$ 的功能描述,它包括 $IC_F$ 和 $OC_F$ 两部分.其中 $IC_F$ 是一谓词表达式,表示 $DO_F$ 上的关系,记为 $IC_F(X)$,称为输入断言;凡满足输入断言的输入值称为合法输入.$OC_F$ 是一谓词表达式,表示 $DO_F\times RA_F$ 上的关系,记为 $OC_F(X,Y)$,称为输出断言.给定合法输入 $X'$,凡满足 $OC_F(X',Y')$的输出 $Y'$称为关于 $X'$的合法输出.如果 $F$ 是功能已知的基元函数,那么相应的 $P(F)$和 $SP_F$ 分别被称为基元函数功能规格说明和基元函数功能描述.

**定义** 1.3. 函数 $F$ 的 HOS 规格说明是一个二元组,即 $H(F)=\langle FD(F),SD_F\rangle$,其中 $FD(F)$是函数 $F$ 的说明,$SD_F$ 是函数 $F$ 的算法描述,它指的是:(1)$SD_F$ 是基元函数标记 $p$;或(2)$SD_F$ 是递归标记 $r$;或(3)$SD_F$ 是 $CL(H(F_1),H(F_2))$($CL$ 是前述的各种控制结构)且每一 $H(F_i)(i=1,2)$是函数 $F_i$ 的 HOS 规格说明.

**定义** 1.4. 给定函数功能规格说明 $P(F)$,HOS 规格说明 $H(F)$是 $P(F)$的解是指对于满足输入断言的任一合法输入 $X$,如果 $H(F)(X)$的计算终止,则其输出关于 $X$ 是合法的,即$\forall\ X\in DO_F(IC_F(X)\rightarrow OC_F(X,H(F)(X)))$.

**定义** 1.5. HOS 规格说明的功能理解是指由 $H(F)$构造相应的 $P(F)$并使得 $H(F)$是 $P(F)$的解.

**定义** 1.6. 在一阶理论中,设有两公式$\forall\ x_1,\cdots,x_n[P\rightarrow Q]$和 $A(x_1,\cdots,x_i)$,其中 $A(x_1,\cdots,x_i)$仅包含自由变量 $x_1,\cdots,x_i(1\leqslant i\leqslant n)$,公式 $A(x_1,\cdots,x_i)$是$\forall\ x_1,\cdots,x_n[P\rightarrow Q]$的$\{x_1,\cdots,x_i\}$一前件当且仅当$\forall\ x_1,\cdots,x_n[P\rightarrow(A(x_1,\cdots,x_i)\rightarrow Q)]$成立.在实际应用中,一般要求所导出的前件尽可能的弱和简单.

类似地,可定义后件的概念并可把后件推导化为前件推导问题[3].Smith[4]提出了导出前件的形式系统,我们已在 SUN 工作站上实现了这一系统[5].前件推导与后件推导机制是本文所采用的基本机制.

## 2　语义构造规则

### 2.1　基本结构语义构造规则

基本结构包括 OR 结构、INCLUDE 结构和 JOIN 结构,相应规则分别被称为 OR 规则、INCLUDE 规则和 JOIN 规则.

#### 2.1.1　OR 规则

OR 结构可表示如图 2.其含义是子函数 $F_1$ 和 $F_2$ 的输入均恒同于父函数 $F$ 的输入;子函数 $F_1$ 和 $F_2$ 的输出均恒同于父函数 $F$ 的输出;其执行是根据 $B(X)$的真假从 $F_1$ 和 $F_2$ 中择一执行.

OR 规则如下:

FROM:$\langle Y=F_1(X),Pre_1(X),Post_1(X,Y)\rangle$ WITH $B(X)$

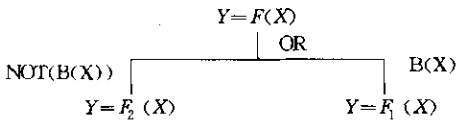



图2

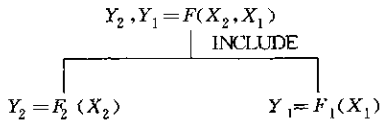



图3

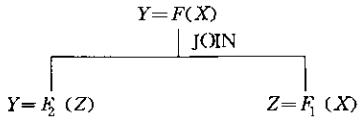



图4

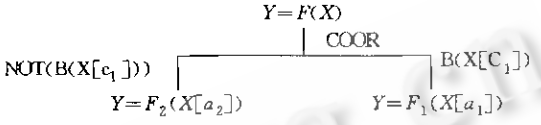



图5

$\langle Y=F_2(X), Pre_2(x), Post_2(X,Y)\rangle$ WITH NOT$(B(X))$

COND: TRUE

GENE: $\langle Y=F(X), [Pre_1(X)\wedge B(X)]\vee[Pre_2(X)\wedge NOT(B(X))],$

$[Post_1(X,Y)\wedge B(X)]\vee[Post_2(X,Y)\wedge NOT(B(X))]\rangle$

其中 FROM 表示已知条件,COND 表示假设条件,GENE 表示所产生的结果,下同.

2.1.2 INCLUDE 规则

INCLUDE 结构可表示如图 3. 其含义是子函数 $F_1$ 的输入与输出分别恒同于父函数 $F$ 输入与输出的第二部分;子函数 $F_2$ 的输入与输出分别恒同于父函数输入与输出的第一部分;$F_1$ 和 $F_2$ 均需执行.

INCLUDE 规则如下:

FROM: $\langle Y_1=F_1(X_1), Pre_1(X_1), Post_1(X_1,Y_1)\rangle$

$\langle Y_2=F_2(X_2), Pre_2(X_2), Post_2(X_2,Y_2)\rangle$

COND: TRUE

GENE: $\langle Y_1,Y_2=F(X_1,X_2), Pre_1(X_1)\wedge Pre_2(X_2), Post_1(X_1,Y_1)\wedge Post_2(X_2,Y_2)\rangle$

2.1.3 JOIN 规则

JOIN 结构可表示如图 4. 其含义是子函数 $F_1$ 的输入恒同于父函数 $F$ 的输入,子函数 $F_2$ 的输入恒同于子函数 $F_1$ 的输出;子函数 $F_2$ 的输出恒同于父函数 $F$ 的输出;其执行过程是先执行 $F_1$,再执行 $F_2$.

JOIN 规则如下:

FROM: $\langle Z=F_1(X), Pre_1(X), Post_1(X,Z)\rangle$

$\langle Y=F_2(Z), Pre_2(Z), Post_2(Z,Y)\rangle$

CONS: $\forall X,Z[Pre_1(X)\wedge Post_1(X,Z)\rightarrow Pre_2(Z)]$

GENE: $\langle Y=F(X), Pre_1(X), \forall X,Y,Z[Pre_1(X)\wedge Post_1(X,Z)\wedge Post_2(Z,Y)]$的$\{X,Y\}$—后件$\rangle$

## 2.2 复合结构语义构造规则

由基本控制结构定义的复合控制结构对父函数中输入输出变量在子函数中的划分具有更大的灵活性,使用起来也更加方便.

复合控制结构包括 COOR、COINCLUDE 和 COJOIN,相应的规则分别被称为 COOR 规则、COINCLUDE 规则和 COJOIN 规则.

2.2.1　COOR 规则

COOR 结构可表示如图 5. 其含义是子函数 $F_1$ 与 $F_2$ 的输入均可以是父函数输入的子集,而子函数 $F_1$ 和 $F_2$ 的输出均恒同于父函数的输出;其执行过程同 *OR* 结构(其中 $X[a_i]$, $X[c_i]$表示 $X$ 的子集,下同).

COOR 规则如下:

FROM:$\langle Y=F(X[a_1]),Pre_1(X[a_1]),Post_1(X[a_1],Y)$ WITH $B(X[c_1])\rangle$

$\langle Y=F(X[a_2]),Pre_2(X[a_2]),Post_2(X[a_2],Y)$ WITH NOT$(B(X[c_1]))\rangle$

COND:TRUE

GENE:$\langle Y=F(X),[Pre_1(X[a_1])\wedge B(X[c_1])]\vee[Pre_2(X[a_2])\wedge \mathrm{NOT}(B(X[c_1])],$

$[Post_1(X[a_1],Y)\wedge B(X[c_1])]\vee[Post_2(X[a_2],Y)\wedge \mathrm{NOT}(B(X[c_1])]\rangle$

2.2.2　COINCLUDE 规则

COINCLUDE 结构如图 6. 其含义是,子函数 $F_1$ 和 $F_2$ 的输入均可以是父函数 $F$ 输入的子集;而 $F_1$ 和 $F_2$ 的输出则分别恒同于父函数输出的第二部分和第一部分;其执行过程同 INCLUDE.

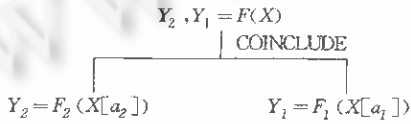



图6

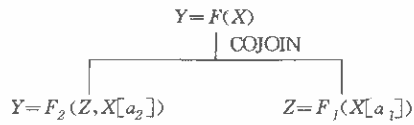



图7

COINCLUDE 规则如下:

FROM:$\langle Y_1=F_1(X[a_1]),Pre_1(X[a_1]),Post_1(X[a_1],Y_1)\rangle$

$\langle Y_2=F_2(X[a_2]),Pre_2(X[a_2]),Post_2(X[a_2],Y_2)\rangle$

COND:TRUE

GENE:$\langle Y_1,Y_2=F(X),Pre_1(X[a_1])\wedge Pre_2(X[a_2]),Post_1(X[a_1],Y_1)\wedge Post_2(X[a_2],$

$Y_2)\rangle$

2.2.3　COJOIN 规则

COJOIN 结构可表示如图 7. 其含义是,子函数 $F_1$ 的输入可以是父函数输入的子集;子函数 $F_2$ 的输入由子函数 $F_1$ 的输出与父函数输入的子集构成,而子函数 $F_2$ 的输出则需恒同于父函数 $F$ 的输出;其执行过程同 JOIN.

COJOIN 规则如下:

FROM:$\langle Z=F_1(X[a_1]),Pre_1(X[a_1]),Post_1(X[a_1],Z)\rangle$

$\langle Y=F_2(Z,X[a_2]),Pre_2(Z,X[a_2]),Post_2((Z,X[a_2]),Y)\rangle$

COND:TRUE

GENE:$\langle Y=F(X),Pre_1(X[a_1])\wedge[Pre_1(X[a_1])\wedge Post_1(X[a_1],Z)\rightarrow$

$Pre_2(Z,X[a_2])]$的$\{X\}$—前件(记为 $PreF(X)$),

$PreF(X)\wedge Post_1(X[a_1],Z)\wedge Post_2((Z,X[a_2]),Y)$的$\{X,Y\}$—后件$\rangle$

**2.3　规则的正确性**

所谓规则是正确的是指,如果 $H(F_1)$和 $H(F_2)$分别是子函数功能规格说明 $P(F_1)$与 $P$

$(F_2)$的解，那么，$CL(H(F_1),H(F_2))$是由相应规则所构造出的父函数功能规格说明$P(F)$的解. 可以证明，上面所给出的语义构造规则是正确的. 由于上述的正确性准则具有传递性，因此，基于各种语义构造规则所构作的软件功能规格说明一定满足软件功能理解的定义.

例如，在JOIN规则的条件下，如果$H(F_1)$和$H(F_2)$分别是$P(F_1)$与$P(F_2)$的解，则JOIN$(H(F_1),H(F_2))$是由JOIN规则所构造出的$P(F)$的解. 证明如下：

由$H(F_1)$是$P(F_1)$的解可得

$$\forall\ X\in DO_{F1}[Pre_1(X)\rightarrow Post_1(X,H(F_1)(X))] \tag{1}$$

由$H(F_2)$是$P(F_2)$的解可得

$$\forall\ Z\in RA_{F1}[Pre_2(Z)\rightarrow Post_2(Z,H(F_2)(Z))] \tag{2}$$

其中

$$H(F_1)(X)\in RA_{F1},H(F_2)(X)\in RA_{F2} \tag{3}$$

记$\forall\ X,Y,Z[Pre_1(X)\wedge Post_1(X,Y)\wedge Post_2(X,Y)]$的$\{X,Y\}$—后件为$Post(X,Y)$；

现要证下式成立：

$$\forall\ X\in DO_{F1}[Pre_1(X)\rightarrow Post(X,JOIN(H(F_1),H(F_2))(X)).$$

任取$X'\in DO_{F1}$，由JOIN的计算规则和(1)可得，有$Z'=H(F_1)(X')$使得$Post_1(X',Y')$成立；由于$Z'\in RA_{F1}$，再由JOIN计算规则和(2)可得，有$Y'=H(F_2)(Z')$使得$Post_2(Z',Y')$成立.

由$Pre_1(X')\wedge Post_1(X',Z')\wedge Post_2(Z',Y')$成立和后件定义可得$Post(X',Y')$成立. 由于$X'$是任意的且$Y'=$JOIN$(H(F_1),H(F_2))(X')$，所以定理得证.

## 3 层次式功能理解与验证

一般说来，程序理解与程序验证密切相关. 程序理解过程在某种意义上可看作程序验证过程；而形式的理解方法必然导致严格的程序验证.

与HOS软件规格说明的设计过程相反，HOS软件规格说明的理解过程由一系列自底向上的理解步构成；每一理解步由两个子函数的功能描述根据相应的结构规则构造出父函数的功能描述. 这种理解过程从HOS软件规格说明树的叶结点开始. 一直进行到构造出根结点的功能描述为止.

为了实现HOS软件规格说明的分步验证，我们将对HOS规格说明树中具有相对独立功能的函数设置验证点，验证点有两类，其一是根结点，其二是被递归的分支结点；对于每一验证点，要求用户给出相应函数的功能描述，然后利用上述理解过程实现HOS软件规格说明的正确性验证. 具体步骤如下：

a. 设置验证点

分析HOS规格说明树的结构，找出根结点和被递归的分支结点，由用户给出相应的功能描述$\langle Pre(X),Post(X,Y)\rangle$和$\langle Pre_i(X_i,Post_i(X_i,Y_i)\rangle$.

b. 置换代入叶结点功能

对非递归的叶结点，由基本函数的知识库获取其功能描述；而对于递归叶结点，由用户给出的验证点功能描述获取相应的功能描述.

c. 逐层理解

从 HOS 规格说明树的叶结点开始,逐层构造分支结点的功能描述.

d. 分步验证

对每一验证点,验证由逐层理解所生成的函数功能描述$\langle Pre_i'(X_i), Post_i'(X_i,Y_i)\rangle$构成用户所给出的相应函数功能描述$\langle Pre_i(X_i), Post_i(X_i,Y_i)\rangle$的精化(refinement),即

$$\forall\ X\in DO_{Fi}(Pre_i(X_i)\rightarrow Pre_i'(X_i))$$

$$\forall\ X\in DO_{Fi}, Y\in RA_{Fi}(Pre_i(X_i)\wedge Post_i'(X_i,Y_i)\rightarrow Post_i(X_i,Y_i))$$

结果,如果对每一验证点精化条件均成立,则相应的 HOS 规格说明满足用户的要求,否则,它可能是不正确的.

例:

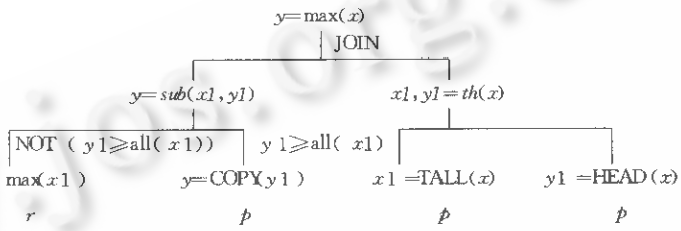
a. HOS 规格说明

FUNCTION

DECLARE

$x, x1: SEQ(INT)$;

$y, y1: INT$;




b. 设置验证点

验证点即为根结点,所希望的功能描述为　　$\langle x\neq\Phi,\ y\geqslant all(x)\wedge y\in x\rangle$

c. 生成叶结点功能描述

$y1=HEAD(X)$　　$[p]$　　$\langle x\neq\Phi,\ y1=HEAD(x)\rangle$

$x1=TAIL(X)$　　$[p]$　　$\langle TRUE,\ x1=TAIL(x)\rangle$

$y=COPY(y1)$　　$[p]$　　$\langle TRUE,\ y=y1\rangle$

$y=\max(x1)$　　$[r]$　　$\langle x1\neq\Phi,\ y\in x1\wedge y\geqslant all(x1)\rangle$

d. 逐层理解

使用 COINCLUDE 规则,可生成函数 $x1, y1=th(X)$的功能描述如下:

$$\langle x\neq\Phi, y1=HEAD(x)\wedge x1=TAIL(x)\rangle$$

使用 COOR 规则,可生成函数 $y=sub(x1,y1)$的功能描述如下:

其输入断言为　　$(TRUE\wedge y1\geqslant all(x1))\vee(NOT(y1\geqslant all(x1)\wedge x1\neq\Phi)$

此式可简化为　　$y1\geqslant all(x1)\vee x1\neq\Phi$;

而其输出断言如下:

$$(y=y1\wedge y1\geqslant all(x1))\vee(NOT(y1\geqslant all(x1))\wedge(y\in x1\wedge y\geqslant all(x1))$$

由 JOIN 规则,其前提条件 COND 显然成立;结果,对函数 $y=\max(x)$所构造出的功能描述如下:

输入断言:$x\neq\Phi$

输出断言:$x\neq\Phi\wedge(y1=HEAD(x)\wedge x1=TAIL(x))\wedge((y=y1\wedge y1\geqslant all(x1))$

$\vee(NOT(y1\geqslant all(x1))\wedge(y\in x1\wedge y\geqslant all(x1))$的$\{x,y\}$—后件,

即

$$x\neq\Phi\wedge y\in x\wedge y\geqslant all(x)$$

e. 精化条件成立.

## 4 自动复用

实现HOS软件规格说明的自动复用要解决两方面的技术问题;其一是如何自动获取用于刻划HOS规格说明功能的函数功能规格说明;其二是对待解的函数功能规格说明,如何进行语义功能匹配,以便在库中正确地获取可复用的HOS规格说明.

对于第一个问题,在受限的情形下可部分的得到解决.基本思想是,如果所给的HOS规格说明不包含递归结点,则可通过层次式功能理解方法自动获取相应的功能规格说明;如果所给的HOS规格说明仅包含递归于根的叶结点,则可根据HOS规格说明可执行的特征由运行方法获取其输入输出的正负实例集,然后通过归纳学习方法来猜测相应的功能规格说明[6],最后用层次式功能理解方法对其正确性进行验证.

对于第二个问题,可定义函数功能规格说明间的可满足性关系,以此作为在库中获取相应HOS规格说明的准则,并利用自动演绎机制对其进行自动验证.

设两给定的函数功能规格说明分别是

$$P_1(F)=\langle\langle Y=F(X),DO_F,RA_F\rangle,\langle IC_{F1}(X),OC_{F1}(X,Y)\rangle\rangle$$

$$P_2(F)=\langle\langle Y=F(X),DO_F,RA_F\rangle,\langle IC_{F2}(X),OC_{F2}(X,Y)\rangle\rangle$$

如果以下两式成立:

$$\forall\ X\in DO_F(IC_{F2}(X)\rightarrow IC_{F1}(X))$$

$$\forall\ X\in DO_F,Y\in RA_F(IC_{F1}(X)\wedge OC_{F2}(X,Y)\rightarrow OC_{F1}(X,Y))$$

则称$P_2(F)$满足$P_1(F)$.

可以证明,$P_2(F)$的任一解也是$P_1(F)$的解;因此,设待解的函数功能规格说明为$P_1(F)$,则在库中寻找正确的可复用的HOS规格说明就是通过适当的换名在库中查找其功能规格说明满足$P_1(F)$的HOS规格说明.可满足性的验证可由前件推导机制自动完成.

## 5 结 语

HOS规格说明的功能理解实际上是一逆向工程(reverse engineering)的问题[7].本文基于前件和后件推导的概念,提出了一种能在某种意义下保证HOS规格说明语义正确性的层次式功能理解方法并初步讨论其在HOS规格说明自动复用中的应用.

进一步的工作包括两个方面.一方面,定义功能更强的多叉树形结构并研究其语义构造规则,使此方法能适合较大规格说明;另一方面,对可满足性条件不成立的情形,研究对其相应HOS规格说明的某些部分加以自动修正的规则,使HOS规格说明自动复用问题的研究更进一步.

**致谢** 本项研究工作是在徐家福教授的指导下完成的,课题组的其他同志参加了讨论并提出了有益的建议,在此表示衷心感谢.

### 参考文献

1 Hamilton M, Zeldin S. Higher order software—a methodology for defining software. IEEE. Trans. on Software

Engineering, 1976,2(1):9—32.
2 Hamilton M, Zeldin S. The functional life cycle model and its automation: use it. Journal of Systems and Software, 1983,3(1):25—62.
3 朱迎春. NDSAIL 系统中的有向推理机制[硕士论文]. 南京大学,1992.
4 Smith D R. Derived preconditions and their use in program synthesis. Lecture Notes in Computer Science, 138.
5 吕建. 算法设计自动化系统 NDADAS 的设计与实现[博士论文]. 南京大学,1988.
6 Lu Jian. A method of acquiring formal specifications from examples. ACM Software Engineering Notes, 1992,17(1):53—57.
7 McClure C. The three Rs of software automation. Prentice Hall, Englewood Cliffs, 1992.

# THE FUNCTIONAL UNDERSTANDING OF HOS SPECIFICATION AND ITS APPLICATION

Lü Jian　Fei Zongming

(*Institute of Computer Software, Nanjing University, Nanjing* 210093)

**Abstract**　This paper gives the semantic synthesis rules for HOS's primitive structures JOIN、INCLUDE、OR and co—structures COINCLUDE、COOR、COJOIN. Based on these semantic rules, a hierarchical understanding method for HOS specification is presented and its applications to the verification and reuse of HOS specification are also discussed.

**Key words**　HOS specification, functional understanding, verification, reuse.