XYZ/SE 程序的验证*

张文辉

(中国科学院软件研究所,北京 100080)

摘要 XYZ/E 的好处之一在于高级和低级的说明能够在同一框架下表示,因而使得软件的说明和实现变得容易一些.在这同时,开发验证工具以验证不同层次的说明是否满足所期望的关系是很重要的.谢洪亮等同志曾研究过 XYZ/SE 程序的验证规则.本篇文章增加了有关使用数组、过程说明和过程调用的规则.同时着重说明 XYZ/SE 程序验证的自动化方面的问题,且实现了一些化简验证条件的规则.

关键词 时序逻辑,程序验证,XYZ 系统.

XYZ/E 是一个程序语言,同时也是一个时序逻辑系统[1].它的好处之一在于高级和低级的说明能够在同一框架下表示.因而使得软件的说明和实现变得容易一些.在这同时,开发验证工具以验证不同层次的说明是否满足所期望的关系是很重要的(比如,给出说明 A 和 C,验证 $C \Rightarrow A$ 是否成立).在这篇文章里,我们选择结构化 XYZ/E(即 XYZ/SE[2])的一个子集作为验证的对象.我们讨论 XYZ/SE 语言中的条件元、串合成、简单选择语句(两种选择)、循环语句和过程调用.

· 条件元的形式为:$LB=l1 \Rightarrow \$OLB=l2 \wedge \$Ox_1=e_1 \wedge \cdots \wedge \$Ox_n=e_n$

它的作用就是把值$(e_1,\cdots,e_n)$赋给$(x_1,\cdots,x_n)$.每段 XYZ/SE 程序都有一入口标记和一出口标记.我们就用这两个标记作为这一段程序的标记.在上面的这段程序中,入口标记为 $l1$,出口标记为 $l2$.通常,我们用 $b(l1,l2)$表示入口标记为 $l1$,出口标记为 $l2$ 的一段程序.

· 串合成的形式为:$b(l1,l2);b(l2,l3)$

它表示执行程序段 $b(l1,l2)$后接着执行 $b(l2,l3)$.

· 简单选择语句的形式为:

$$?\ [LB=l1 \wedge c \Rightarrow \$OLB=l3 \mid \sim \Rightarrow \$OLB=l4;b(l3,l2);b(l4,l2);]$$

它等价于下面这一段 XYZ/BE[1]程序:

$$LB=l1 \wedge c \Rightarrow \$OLB=l3;LB=l1 \wedge \sim c \Rightarrow \$OLB=l4;b(l3,l2);b(l4,l2);$$

· 循环语句的形式为:

$$*\ [LB=l1 \wedge c \Rightarrow (\$OLB=l3 \mid \$OLB=l2)b(l3,l1);]$$

它等价于下面这一段 XYZ/BE 程序:

---

* 本文 1994-05-12 收到,1994-10-24 定稿

本项目受王宽城基金会的资助.

作者张文辉,1963 年生,副教授,主要研究领域为定理证明,程序验证.

本文通讯联系人:张文辉,北京 100080,中国科学院软件研究所

© 中国科学院软件研究所 http://www.jos.org.cn

$$LB=l1\wedge c\Rightarrow \$OLB=l3;LB=l1\wedge\sim c\Rightarrow \$OLB=l2;b(l3,l1);$$

· 过程调用的形式为:$LB=l1\Rightarrow \$OLB=l2\wedge p(\cdots)$其中 p 为一过程.

一段 XYZ/SE 程序对应于一段 XYZ/BE 程序,而一段 XYZ/BE 程序可被转换成一段 XYZ/SE 程序[3]. 因此验证 XYZ/SE 程序的工具也能用于验证 XYZ/BE 程序.

谢洪亮等同志曾在构造结构化的 XYZ/E(即 XYZ/SE)上做过工作,并且给出了一些类似于 Hoare 公式的验证规则[2]. 这些规则可用于条件元、串合成、选择语句和循环语句. 一个建立在这基础上的系统曾在 ML 上实现[4]. 若要证明一个程序,我们用较小的程序部分和他们的前断言和后断言来组成较大的程序. 因为我们需把一个程序分成最小的部分,并加上他们的前断言和后断言,所以这个系统很难用. 在这里,我们用一个部分自动的策略,即我们只需给出程序、它的前断言、后断言和它的循环语句不变量,则可自动生成验证条件. 从这些条件的正确性可以推出程序的正确性. 并且我们增加了有关使用数组、过程说明和过程调用的规则. 同时,我们也实现了一些化简验证条件的规则. 这个系统是在 SUN 工作站和 X 窗口系统上实现的.

1 程序验证

为描述验证规则,除了前面见到的时序算子 $\$O$ 外,我们需要 $\Box$ 和 $\triangleright$. 若 A 是一个公式,$\$OA$ 表示 A 在下一时刻为真. $\Box A$ 表示 A 在当前和将来都为真. $\triangleright$ 则相当于 Kroger 的 atnext 算子[5].

1.1 基本规则

对于条件元、串合成、选择语句和循环语句,我们可以用文献[2]提供的规则. 为了配合验证策略和实现上的方便,我们对这些规则做了一些修改,并且增加了对替换数组元素的一个解释. 下面是这些规则:

条件元: $\vdash \Box b(l1,l2)\wedge LB=l1\wedge R[e_1/x_1,\cdots,e_n/x_n]\Rightarrow R\triangleright LB=l2$

在这里,$b(l1,l2)$代表 $LB=l1\Rightarrow \$OLB=l2\wedge \$Ox_1=e_1\wedge\cdots\wedge \$Ox_n=e_n$

若 x_i 是一个数组元素,e_i/x_i 就不是一个普通的替换. 在这里,一个数组有一个名称(例如 a)和一串赋值. 这一串中的每一个元素是一串同时赋值. 用 S 表示这样的一串赋值,那么这个数组可表示为 $a[S]$. 让 S 的第一个元素为 L,S 的剩余部分为 S'. 求 $a[S]$的值,就是先用 L(一串同时赋值)给 a 赋值,然后再把 S' 应用在 $a[L]$上.

在 $R[e_1/x_1,\cdots,e_n/x_n]$中,若$(x_{i_1},\cdots,x_{i_k})$(其中 $i_1,\cdots,i_k$ 是集合$\{1,\cdots,n\}$中的元素)是$(a[t_{11},\cdots,t_{1m}],\cdots,a[t_{k1},\cdots,t_{km}])$,我们把 $a[S]$(假设它出现在 R 内)替换成

$$a[[(t_{11},\cdots,t_{1m})\rightarrow e_{i_1},\cdots,(t_{k1},\cdots,t_{km})\rightarrow e_{i_k}],S].$$

在此$(t_{j1},\cdots,t_{jm})\rightarrow e$ 表示把值 e 赋给 $a[t_{j1},\cdots,t_{jm}]$.

串合成:

$$\frac{\vdash \Box b(l1,l3)\wedge LB=l1\wedge R1\Rightarrow R2\triangleright LB=l3 \quad \vdash \Box b(l3,l2)\wedge LB=l3\wedge R2\Rightarrow R3\triangleright LB=l2}{\vdash \Box b(l1,l2)\wedge LB=l1\wedge R1\Rightarrow R3\triangleright LB=l2}$$

在这里,$b(l1,l2)$代表 $b(l1,l3);b(l3,l2)$.

简单选择语句(两种选择):

© 中国科学院软件研究所 http://www.jos.org.cn

$$\frac{\begin{array}{c}\vdash \Box b(l3,l2)\wedge LB=l3\wedge R1\Rightarrow R2\triangleright LB=l2\\ \vdash \Box b(l4,l2)\wedge LB=l4\wedge R3\Rightarrow R2\triangleright LB=l2\end{array}}{\vdash \Box b(l1,l2)\wedge LB=l1\wedge(c\rightarrow R1)\wedge(\sim C\rightarrow R3)\Rightarrow R2\triangleright LB=l2}$$

在这里 $b(l1,l2)$表示

$$?\,[LB=l1\wedge c\Rightarrow \$OLB=l3\,|\sim\Rightarrow \$OLB=l4;b(l3,l2);b(l4,l2);]$$

循环语句:

$$\frac{\begin{array}{c}\vdash \Box b(l3,l1)\wedge LB=l3\wedge R1\Rightarrow R\triangleright LB=l1\\ \vdash \Box(R\wedge c\rightarrow R1)\\ \vdash \Box(R\wedge\sim c\rightarrow R')\end{array}}{\vdash \Box b(l1,l2)\wedge LB=l1\wedge R\Rightarrow R'\triangleright LB=l2}$$

在这里 $b(l1,l2)$表示

$$*\,[LB=l1\wedge c\Rightarrow(\$OLB=l3\,|\,\$OLB=l2)b(l3,l1);]$$

在规则的实现上,$R\wedge C\rightarrow R1$ 和 $R\wedge\sim C\rightarrow R'$ 成了验证条件.

1.2　过程调用规则

验证包含过程调用的程序,方法之一就是用过程的代码替换过程调用.若这样做,其中的一个问题是这方法不能用在递归调用上.另外,若一个过程用在不同的地方,则需多个类似的证明.这会降低系统的工作效率.我们把过程调用当作一个广义赋值.我们有两个规则,一个有关过程说明,一个有关过程调用.

过程说明规则:

$$\frac{\begin{array}{c}\vdash \Box b(l1,l2)\wedge LB=l1\wedge Q\Rightarrow R\triangleright lb=l2\\ \vdash \Box(Q'\rightarrow Q)\end{array}}{\vdash \Box b(l1',l2')\wedge LB=l1'\wedge Q'\Rightarrow R\wedge B\triangleright lb=l2'}$$

以下是这个规则的解释:

1. $b(l1',l2')$是

$LB=l1'\Rightarrow \$OLB=l2'\wedge p(\%IOP/y_1\,|\,y_1;\cdots;\%IOP/y_n\,|\,y_n;\%INP/y_{n+1}\,|\,y_{n+1};\cdots;\%INP/y_{n+m}\,|\,y_{n+m})$

2. p 是一个过程,它的说明如下:

$\%PROC\ p(\%IOP/y_1:T;\cdots;\%IOP/y_n:T;\%INP/y_{n+1}:T;\cdots;\%INP/y_{n+m}:T)$

$==$

$\%LOC[\cdots]$

$\%STM[b(l1,l2)]$

在规则的实现上,没用到局部变量说明中的信息,我们假设局部变量名不出现在参数表,前断言和后断言中.

3. B 的形式为 $X_1=@X_1\wedge\cdots\wedge X_k=@X_k$

其中$@X_1,\cdots,@X_k$是新符号.我们用 $X=@X$ 表示 X 的值为$@X$. B 不需证明.在规则的实现上,我们把 $Q'\rightarrow Q$ 作为验证条件.

过程调用规则

$$\frac{\vdash \Box b(l1,l2)\wedge LB=l1\wedge P\wedge A\Rightarrow R\wedge B\triangleright LB=l2}{\vdash \Box b(l1',l2')\wedge LB=l1'\wedge P'\wedge(R'\rightarrow Q')\Rightarrow Q\triangleright LB=l2'}$$

以下是这个规则的解释:

© 中国科学院软件研究所 http://www.jos.org.cn

1. $b(l1', l2')$是

$LB=l1' \Rightarrow \$OLB=l2' \wedge p(\%IOP/x_1 \mid y_1;\cdots;\%IOP/x_n \mid y_n;\%INP/t_1 \mid z_1;\cdots;\%INP/t_m \mid z_m)$

2. $b(l1, l2)$是

$LB=l1 \Rightarrow \$OLB=l2 \wedge p(\%IOP/y_1 \mid y_1;\cdots;\%IOP/y_n \mid y_n;\%INP/z_1 \mid z_1;\cdots;\%INP/z_m \mid z_m)$

3. B是 $y_1=e_1 \wedge \cdots \wedge y_n=e_n \wedge y_{n+1}=e_{n+1} \wedge \cdots \wedge y_{n+l}=e_{n+l}$.

其中 $y_{n+1},\cdots,y_{n+l}$是在 p 中所有受到改变的全局变量. 我们对 B 的要求是由 B 不能推导出形如 $V=\cdots V\cdots$的等式. 若我们不需或不能表示某变量(如 V)的值,则可用 $V=@V$ 来代替. 在此,$@V$ 并非一个常量. 每次使用这个规则时,我们都得用一新符号更换这里的$@V$.

4. A是 $y_{i_1}=\#y_{i_1} \wedge \cdots \wedge y_{i_j}=\#y_{i_j} \wedge y_{i_{j+1}}=\#y_{i_{j+1}} \wedge \cdots \wedge y_{i_k}=\#y_{i_k}$

其中 $i_1,\cdots,i_j$是$\{1,\cdots,n\}$中的元素,$i_{j+1},\cdots,i_k$是$\{n+1,\cdots,n+l\}$中的元素. $\#y_i$是用来表示 y_i 初始值的新符号.

5.

$$P''=P[x_1/y_1,\cdots,x_n/y_n,t_1/z_1,\cdots,t_m/z_m]$$
$$P'=P''[x_{i_1}/\#y_{i_1},\cdots,x_{i_j}/\#y_{i_j},y_{i_{j+1}}/\#y_{i_{j+1}},\cdots,y_{i_k}/\#y_{i_k}]$$
$$R''=R[e_1'/y_1,\cdots,e'_{n+l}/y_{n+l}][t_1/z_1,\cdots,t_m/z_m]$$
$$R'=R''[x_{i_1}/\#y_{i_1},\cdots,x_{i_j}/\#y_{i_j},y_{i_{j+1}}/\#y_{i_{j+1}},\cdots,y_{i_k}/\#y_{i_k}]$$
$$Q''=Q[e_1''/x_1,\cdots,e_n''/x_n,e''_{n+1}/y_{n+1},\cdots,e''_{n+l}/y_{n+l}]$$
$$Q'=Q''[x_{i_1}/\#y_{i_1},\cdots,x_{i_j}/\#y_{i_j},y_{i_{j+1}}/\#y_{i_{j+1}},\cdots,y_{i_k}/\#y_{i_k}].$$

其中 e'_i是用 B 中的等式将 e_i 中的 $y_j(1\leqslant j\leqslant n+l)$全替换掉后的结果. $e''_i=e'_i[t_1/z_1,\cdots,t_m/z_m]$.

例如 1:在前断言中用 $x=\#x$ 等式的例子. 过程 f 的前断言是 $a=\#a$,后断言是 $a=+(\#a,1)$. 若不用这种形式,就难以说明后断言,比如,若用 $a=a_0$ 作前断言,用 $a=+(a_{0,1})$作后断言,则会使得过程 g 的正确性无法证明. 在规则的实现上,我们用#作为这种常量的名称的第一个字母.

$\{a=\#a\}$

$\%PROC\ f(\%IOP/a;INT)==$

$\%STM[LB=START \Rightarrow \$OLB=RETURN \wedge \$Oa=+(a,1);]$

$\{a=+(\#a,1)\}$

$\{b=0\}$

$\%PROC\ g(\%IOP/b;INT)==$

$\%STM[LB=START \Rightarrow \$OLB=RETURN \wedge f(\%IOP/b \mid a);]$

$\{b=1\}$

过程 g 中 $b(START,RETURN)$的前断言为$(b=1)[+(\#a,1)/b][b/\#a]$即$+(b,1)=1$. 我们得到的验证条件就是 $b=0\rightarrow+(b,1)=1$.

例如 2:在后断言中用形如 $x=@x$ 等式的例子. 这个例子的目的在于说明这种等式的用法和必要性.

$\{T\}$

© 中国科学院软件研究所 http://www.jos.org.cn

$\%PROC\ f(\%IOP/a:INT)==$

$\%STM[LB=START\Rightarrow \$OLB=RETURN \wedge \$Oa=1;]$

$\{a>0 \wedge a=@a\}$

$\{b>1\}$

$\%PROC\ g(\%IOP/b:INT)==$

$\%STM[LB=START\Rightarrow \$OLB=RETURN \wedge f(\%IOP/b|a);]$

$\{b>1\}$

过程 g 的后断言是不对的.但是如果把 $a=@a$ 从 f 的后断言中去掉.那么由以上的这些证明规则,我们就会得到这个错误的结论.这是因为 g 的后断言中的 b 和前断言中的 b 是不一样的.去掉 f 中的 $a=@a$ 我们就没法区分这两个不一样的 b.

递归调用

以上的规则同样适用于递归调用.我们可以用归纳法来证明这种做法的正确性.我们在递归深度上归纳(若程序不终止,则无需证明).让 $C(k)$ 表示某过程 p 递归调用自己的次数不超过 k.让 Q 表示 p 的前后断言是正确的.我们需证明:$C(0)\rightarrow Q$ 和若 $n>0$ 则 $C(n-1)\rightarrow Q$ 推出 $C(n)\rightarrow Q$.

从应用以上规则证明 p 的正确性的证明中,我们可以得到这样的结论:假使 p 的前后断言的正确的,则 $C(0)\rightarrow Q$ 是正确的.这里要说明的是 $C(0)\rightarrow Q$ 的正确性并不依赖于这个假设.我们分两种情形.

(1)p 出现在某段程序 ? $[LB=l1 \wedge d\Rightarrow \$OLB=l2|\sim\Rightarrow \$OLB=l3;b(l2,l4);b(l3,l4);]$中的 $b(l2,l4)$中,且 d 为假.

(2)p 出现在某段程序 $*[LB=l1 \wedge d\Rightarrow(\$OLB=l2|\$OLB=l3)b(l2,l1);]$的 $b(l2,l1)$中且 d 为假.

在第 1 种情形中,因 d 为假,所以 $b(l2,l4)$的前断言对正确性的证明没影响.因此对 $b(l2,l4)$中的 p 所做的假设是无关紧要的.同样,在第 2 种情形中,对 p 所做的假设也是无关紧要的,所以 $C(0)\rightarrow Q$ 是否正确并不依赖于对 p 所做的假设.

$C(n-1)\rightarrow Q$ 推出 $C(n)\rightarrow Q$,可以从对 p 的正确性证明中得到.

2　简化验证条件

应用上一节的规则证明程序的正确性,我们得到一系列的验证条件.对于简单的验证条件,通过化简,我们可以将它们去掉.剩下的则要用别的办法来证明.

逻辑公式由以下的规则构成:

(1)T 是一个公式.

(2)若 t,s 是项,则 $t=s,t>s,t<s$ 是公式.

(3)若 $t_1,\cdots,t_n$ 是项,p 是一个 n 目谓词,则 $p(t_1,\cdots,t_n)$是一个公式.

(4)若 A 是一个公式,则$\sim A$(非 A),$\forall x:(A)$(对所有 x,A 成立)是公式.

(5)若 A,B 是公式,则 $A\vee B,A\wedge B,A\rightarrow B$ 是公式.

在这里 T 是一个总是为真的逻辑常量.符号=,>,<的解释就是我们通常所理解的等

© 中国科学院软件研究所 http://www.jos.org.cn

于、大于和小于.要简化一个验证条件 $A\rightarrow B$,我们先把它写成 $A\Rightarrow B$,然后使用下列公理和规则:

·公理:

$A1:\Gamma\Rightarrow\Delta,T$ $A2:\Gamma,A\Rightarrow\Delta,A$ $A3:\Gamma,t>t\Rightarrow\Delta$ $A4:\Gamma\Rightarrow\Delta,t=t.$

我们用 Gentzen 规则[6]处理含逻辑符号 $\wedge$,$\vee$,$\rightarrow$,$\sim$的公式.除此之外,我们有下列规则:

$$R1:\frac{\Gamma,s=t\Rightarrow\Delta}{\Gamma\Rightarrow\Delta,s>t,t>s} \qquad R2:\frac{\Gamma,x=t\Rightarrow\Delta,A(t)}{\Gamma,x=t\Rightarrow\Delta,A(x)}$$

在应用这些规则之前,我们先将公式做些简化.首先我们去掉"<"符号(把 $t_1<t_2$ 替换成 $t_2>t_1$),然后使用下列规则:

$t=t$ 替换成 T

$t>t$ 替换成 $\sim T$

$\sim\sim A$ 替换成 A

$T\wedge A,A\wedge T$ 及 $A\wedge A$ 替换成 A

$\sim T\wedge A,A\wedge\sim T$ 及 $A\wedge\sim A$ 替换成 $\sim T$

$T\vee A,A\vee T,A\vee\sim A$ 替换成 T

$\sim T\vee A,A\vee\sim T$ 及 $A\vee A$ 替换成 A

$T\rightarrow A,\sim A\rightarrow A$ 替换成 A

$\sim T\rightarrow A,A\rightarrow T$ 及 $A\rightarrow A$ 替换成 T

$A\rightarrow\sim T,A\rightarrow\sim A$ 替换成 $\sim A$

若 x 不是 p 的自由变量,则把 $\forall x:(p)$ 替换成 p.

对某些特殊函数,我们也实现了一些简化规则.比如,$x-y>0$(在系统中写为 $-(x,y)>0$)改写为 $x>y$,$0>-(x,y)$ 改写为 $y>x$,$-(x,x)$ 改写为 0,$-(+(x,y),y)$ 和 $+(-(x,y)y)$ 改写为 x.若符号 b 的字母顺序排在符号 a 前,则将 $+(a,b)$ 改写为 $+(b,a)$.以后,我们可以增加新的规则以提高系统的功能.

3 例子和讨论

在这一节,我们用 3 个例子来说明系统的功能.系统所需要的是过程、它的前后断言和循环不变量.如果前断言、后断言或不变量没有预先给出,系统则会要求用户提供这些.系统依据这些东西来产生验证条件.

例如 1:以下是一计算最大公约数的过程.这个例子包含条件元、串合成、循环语句和选择语句.我们要求输入 u 和 v 大于 0,输出 w 等于 $Gcd(u,v)$.在此 $Gcd(u,v)$ 表示 u 和 v 的最大公约数.

$\{u>0\wedge v>0\}$

$\%PROC\ gcd(\%INP/u:INT;\%INP/v:INT;\%IOP/w:INT)==$

$\%LOC[x,y:INT]$

$\%STM[$

$LB=START\Rightarrow\$OLB=l1\wedge\$Ox=u\wedge\$Oy=v;$

© 中国科学院软件研究所 http://www.jos.org.cn

```
        *[LB=l1∧~(x=y)⇒$OLB=l2|$OLB=l6)
          ?[LB=l2∧(y<x)⇒$OLB=l4|~⇒$OLB=l5;
            LB=l4⇒$OLB=l1∧$Ox=-(x,y);
            LB=l5⇒$OLB=l1∧$Oy=-(y,x);
          ]
          {x>0∧y>0∧Gcd(x,y)=Gcd(u,v)}
        ]
        LB=l6⇒$OLB=RETURN∧$Ow=y;
]
{(w=Gcd(u,v))}
```

循环语句 $b(l1,l6)$ 的不变量是 $x>0\wedge y>0\wedge Gcd(x,y)=Gcd(u,v)$.

系统在读完程序、前后断言和不变量后,产生以下验证条件:

- $(\neg x=y\wedge((x>0\wedge y>0)\wedge Gcd(x,y)=Gcd(u,v)))$
  $\Rightarrow((\neg y<x\vee((-(x,y)>0\wedge y>0)\wedge Gcd(-(x,y),y)=Gcd(u,v)))$
  $\wedge(y<x\vee((x>0\wedge -(y,x)>0)\wedge Gcd(x,-(y,x))=Gcd(u,v))))$
- $(\neg\neg x=y\wedge((x>0\wedge y>0)\wedge Gcd(x,y)=Gcd(u,v)))\Rightarrow y=Gcd(u,v)$
- $(u>0\wedge v>0)\Rightarrow((u>0\wedge v>0)\wedge Gcd(u,v)=Gcd(u,v))$

前两个条件从循环语句规则中得到,最后一个条件从过程说明规则中得到. 简化之后,我们得到:

- $x>0,y>0,Gcd(x,y)=Gcd(u,v)\Rightarrow Gcd(x,-(y,x))=Gcd(u,v),x>y,x=y$
- $x=y,y>0,y>0,Gcd(y,y)=Gcd(u,v)\Rightarrow y=Gcd(u,v)$
- $x>y,x>0,y>0,Gcd(x,y)=Gcd(u,v)\Rightarrow Gcd(-(x,y),y)=Gcd(u,v),x=y$

要证明这些条件,我们必须应用有关 Gcd 的公理或性质(比如 $Gcd(y,y)=y$). 所以这不能在我们现在的系统中完成.

例如 2:我们用下面的例子来说明包含数组的前后断言和不变量. 下面的程序把一个数组 a 初始化,使得 $a[i]=i(i=0,\cdots,n-1)$.

```
{¬(n<0)}
%PROC init(%INP/n:INT;%IOP/a:ARRAY)==
%LOC[y:INT]
%STM[
        LB=STRAT⇒$OLB=l1∧$Oy=0;
        *[LB=l1∧~(y=n)⇒($OLB=l2|$OLB=RETURN)
            LB=l2⇒$OLB=l1∧$Oa[y]=y∧$Oy=+(y,1);
            {∀k:((¬(k<0)∧(k<y))→a[k]=k)}
        ]
]
{∀k:((¬(k<0)∧(k<n))→a[k]=k)}
```

系统在读完程序、不变量和前后断言后,产生以下验证条件:

© 中国科学院软件研究所 http://www.jos.org.cn

·(¬y=n∧∀ k:((~k<0∧k<y)→a[k]=k))
⇒∀ k:((¬k<0∧k<+(y,1))→a[[(y)→y]][k]=k)
·(¬¬y=n∧∀ k:((~k<0∧k<y)→a[k]=k))
⇒∀ k:((¬k<0∧k<n)→a[k]=k)
·¬n<0⇒∀ k:((~k<0∧k<0)→a[k]=k)

简化之后,剩下验证条件:

∀ k:((¬k<0∧k<y)→a[k]=k)
⇒y=n,∀ k:((¬k<0∧k<+(y,1))→a[[(y)→y]][k]=k)

例3:我们用这个例子来说明过程调用.这个例子计算数的阶乘.它的前断言为 $a\geqslant 0$,后断言为 $r=Fac(a)$.

{¬(a<0)}
%PROC fac(%IOP/r:INT;%INP/a:INT)==
%STM[
?[LB=STRAT∧(a=0)⇒$OLB=l2|~⇒$OLB=l1;
LB=l2⇒$OLB=RETURN∧$Or=1;
LB=l1⇒$OLB=l3∧fac(%IOP/r|r,%INP/-(a,1)|a);
LB=l3⇒$OLB=RETURN∧$Or=*(a,r);
]
]
{r=Fac(a)}

系统产生一个验证条件:

~a<0⇒((~a=0∨1=Fac(a))∧(a=0∨(~--(a,1)<0∧*(a,Fac(-(a,1)))=Fac(a)))).

简化之后,它变成3个比较简单的验证条件.

· T⇒*(a,Fac(-(a,1)))=Fac(a),a=0,0>a
· a=0⇒1=Fac(0)
· 1>a⇒a=0,0>a

将来的工作:以这最后一个例子为例,若要证明第一或第二验证条件,我们需要用到 Fac 的定义或性质.若要证明最后的一个验证条件,我们需要有关自然数的性质,比如1>a等价于0>a∨0=a.在这个特殊的例子中,我们可以加入一条规则把 i>a(i 是一个正整数,a 是一个整数变量)改写为0>a∨a=0∨…∨a=-(i,1)以证明刚才所说的验证条件.但从系统的整体功能和可靠性来讲,我们必须有系统地增加一些规则、公理.将来可以做的工作包括扩充系统或建立一个独立的系统以便更有效地证明这些验证条件.另外从产生验证条件这方面来讲,我们可以扩充系统以处理一个较大的 XYZ/SE 子集(比如包含指针的子集).

致谢 感谢王宽城基金会的资助.同时感谢唐稚松教授和林惠民教授对此项工作的支持、指导和帮助.

© 中国科学院软件研究所 http://www.jos.org.cn

参考文献

1 Tang C. S. A temporal logic language oriented toward software engineering—an introduction to XYZ system(I). Chinese Journal of Advanced Software Research, 1994,**1**(**1**):1—29.

2 Xie Hongliang, Gong Jie, Tang C. S. A structured temporal logic language: XYZ/SE. Journal of Computer Science and Technology. 1991,**6**(**1**):1—10.

3 Gong Jie. A system of automatic transformation from a temporal logic to its structured version. Techn. Rept, IS—AS—XYZ—88—13, 1988.

4 Li Renwei, Zhang Wenhui, He Pei. An introduction to INCAPS system. Journal of Computer Science and Technology. 1993,**8**(**1**):26—37.

5 Kroger F. Temporal logic of programs. Lecture Notes, Techn. Univ. Munich, Inst. for Informatics, Rept. TUM—18521, 1985.

6 Helmut Schwichtenberg. Proof theory: some applications of cut—elimination. Handbook of Mathematical Logic, North—Holland, 1977.

VERIFICATION OF XYZ/SE PROGRAMS

Zhang Wenhui

(*Institute of Software, The Chinese Academy of Sciences, Beijing* 100080)

Abstract　One of the advantages of XYZ/E is that both high level and low level specifications can be represented in the same frame work, so that it makes specification and implementation of software systems easier. It is important to develop verification tools to verify whether the expected relation between different levels of specifications holds. Rules for verification of XYZ/SE programs had been studied by Xie Hongliang et al. in a previous paper. This paper extends the set of the rules to include rules for use of arrays, for procedure specification and for procedure call. It puts the emphasis on the practical and mechanical aspects of the verification of XYZ/SE programs. A set of rules for simplifying verification conditions is also implemented.

Key words　Temporal logic, program verification, XYZ system.

© 中国科学院软件研究所 http://www.jos.org.cn