一种实现程序结构化转换的交互式图形工具*

龚 洁

(中国科学院软件研究所 北京 100080)

摘要 本文介绍交互式图形工具XYZ/BESE,它能将任意时序逻辑语言XYZ/E的子语言XYZ/BE表示的程序转换成结构化程序.如以XYZ/BE作为中间语言,并将这工具与XYZ与系统中源语言转换系统XYZ/CCSS结合起来,即可将任意其它源语言的程序转换成为结构化程序.本系统是用XYZ/E的交互式图形设计工具XYZ/CFC实现的.结构化转换过程用图形表示,可增加其直观性,使这变换易于理解.

关键词 结构化高级语言,结构化程序转换,图形设计工具,时序逻辑语言.

近年来,由于大型软件更新的迫切需要,使软件再工程(reengineering)的研究十分活跃.其中一个重要的方面即使原来的程序转换成用新型程序语言(如ADA,C,COBOL85等)表示的结构化程序.本文试图在这方面提供一种工具.

时序逻辑语言XYZ/E的基本结构是非嵌套命令式的,它在表示图形的语义及产生式系统等方面有其方便之处,但对于习惯高级语言的用户则又往往引起一些非议.针对这种情况,我们在XYZ/E的框架内又定义了一类具有结构化高级语言形式的子语言XYZ/SE.[3,4]本工具即用于实现从任意XYZ/E的非嵌套命令式型程序到用XYZ/SE表示的结构化程序的转换.转换的基本方法与文献[5]中所述相适.

将这工具与XYZ系统中另外的关于源语言转换工具XYZ/CCSS结合起来使用[4],即可达到实现其它语言程序结构化转换的目的.

本系统是用XYZ系统中图形设计工具XYZ/CFC[1,3]实现的.它具有直观性,易于使用户理解.

1 XYZ/BE,XYZ/SE与XYZ图简介

时序逻辑语言XYZ/SE的程序,是由一列单元构成的.而每一单元则由一组条件元的合取式所构成,它具有如下的形式:

$$\Box[ce_1;\ldots;ce_k] \tag{1}$$

此处,符号"□"表示"所有时刻","；"是合取词"∧"的另一表示形式,而每一$ce_i, i=1,\ldots,$

---

* 作者龚洁,女,1964年生,现为美国凤凰城*Motorola*公司研究所博士后、研究人员,主要研究领域为软件工程,硬件形式描述与设计.

本文通讯联系人:唐稚松,北京100080,中国科学院软件研究所

本文1995-03-16收到修改稿

n，则表示一条件元. 所谓条件元，是具有如下形式的一合式公式：

$$LB=y\wedge P\Rightarrow \$\bigcirc(v1,\ldots,vk)=(e1,\ldots,ek)\wedge \$\bigcirc LB=z \tag{2.1}$$

$$LB=y\wedge P\Rightarrow \$\bigcirc(Q\wedge LB=z) \tag{2.2}$$

$$LB=y\wedge P\Rightarrow\Diamond(Q\wedge LB=z) \tag{2.3}$$

此处，“$\$\bigcirc$”“$\Diamond$”为二时序算子，前者表示“下一时刻”，后者表示“终于”；“$\Rightarrow$”是蕴涵词的另一表示形式，“$\wedge$”即通常的合取词；“LB”为一特殊变量，它指向一标号. 为(2)中 y，z 即此标号，“$\Rightarrow$”左边的由 LB 构成的等式中的标号(为 y)称为定义标号，其右边的 LB 构成的等式中的标号(为 z)称为转入标号. 这类由 LB 构成的等式称为控制等式，它表示程序的流向；而(2.2)中如下形式的等式：

$$\$\bigcirc(v1,\ldots,vk)=(e1,\ldots,ek) \tag{3.1}$$

表示如下的合取式：

$$\$\bigcirc v1=e1\wedge\ldots\wedge\bigcirc vk=ek \tag{3.2}$$

其中每一等式“$\$\bigcirc vi=ei$”表式下一时刻 vi 的值等于本时刻表达式 ei 的值，称为赋值等式. 式(2.2)、(2.3)中，“P”“Q”表示一阶逻辑中的谓词，一般说，P 中不出现量词，称为条件，Q 称为动作. (2.1)与(2.2)表示可执行的条件元，(2.3)则表示抽象描述. 事实上，P，Q 即分别相当于前置条件与后续条件. 只要对这样定义的程序中出现的标号，条件及时序变量作一些限制，(如框架公理，条件的一致性及完全性，标号的正则性等)，由(1)式所示形式组成的单元，就既能表示出算法又能表示抽象描述. 只要稍作扩充，就不难表示出常见高级语言中的递归过程调用及进程通讯等机制.

不过，这样的 XYZ/BE 程序虽有很强的表示力，但也存在一个问题，即这样的程序不具有单入口单出口的嵌套结构，且常常是非结构化的. 为了适应用户使用高级语言的习惯，在 XYZ/E 的框架内，定义了一种具有结构化高级语言形式的子语言 XYZ/SE，它是由与常见高级语言中循环语句与分情形语句(其特殊情形的条件语句)及受限制(只许往后转入下一语句)的上述(2)型条件元所对应的转语句构成的，为了保持 XYZ/E 的统一形式，这种语句的嵌套结构与常见高级语言中语句嵌套各有一点差异，它具有“拉平”的形式，并影响其实质. 在文献[3]中详细讨论了这种语句形式的验证规则.

下面(4)、(5)式分别表示 XYZ/SE 语言中循环语句与分情形语句 2 种形式：

$$*[LB=y\wedge R\Rightarrow \$\bigcirc(LB=w\,|\,LB=EXIT)]; \tag{4.1}$$

$$LB=w[\{|Q|\}\ \$\bigcirc LB=y] \tag{4.2}$$

其中“$EXIT$”即表示转入这整个语句后面紧邻的第一个定义符号. 至于(4.2)式则表示，将时序逻辑公式 Q 改写成 XYZ/E 单元的形式以后，再以“$LB=w$”“$LB=y$”分别替换其唯一的入口与出口控制等式. 为简便计，称每一具有形式(4.2)的程序段为一模块，记成 $\$Block[w,y]$.

至于分情形语句，则写成如下形式：

$$\begin{aligned}?\ [&LB=y\wedge P_1\Rightarrow \$\bigcirc LB=w_1;\\ &|P_2\Rightarrow \$\bigcirc LB=w_2;\\ &\ldots\\ &|P_k\Rightarrow \$\bigcirc LB=w_k;\end{aligned}$$

$$LB=w_1\{|Q_1|\}\$\bigcirc LB=EXIT;$$
$$LB=w_2\{|Q_2|\}\$\bigcirc LB=EXIT;$$
$$\cdots$$
$$LB=w_k\{|Q_k|\}\$\bigcirc LB=EXIT] \quad (5)$$

此处“P_1”,...,“P_k”任意 2 个不能同真,而且任一情形下必有一个为真,(最后一“P_k”可以写成“～”表示剩余情形),“|”是“$LB=y\wedge$”的省写. 为了加强可读性,可在每一循环及分情形语句的两端分别加上带标记的括号“whiledo[...]”(或“*[...]”)及“case[...]”(或“?[...]”).

XYZ 图则是上述 XYZ/BE 及 XYZ/SE 的图形表示形式. 其中图 1 中[a],[b],[c],[d]分别对应于(2.1)(或(2.2)),(2.3),(4),(5). 图 1[c]中“‖”表示等待,相当于 Petri 网中的“Transition”,它用于表示通讯进程的同步机制. 此时 R 的真假不决定于本进程而决定于其它进程.

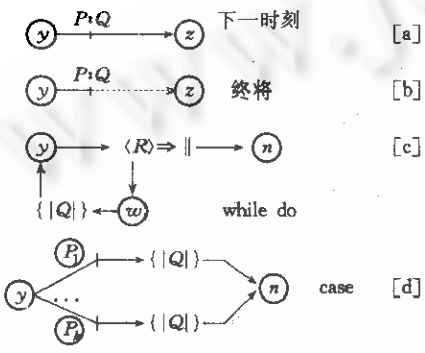


图1

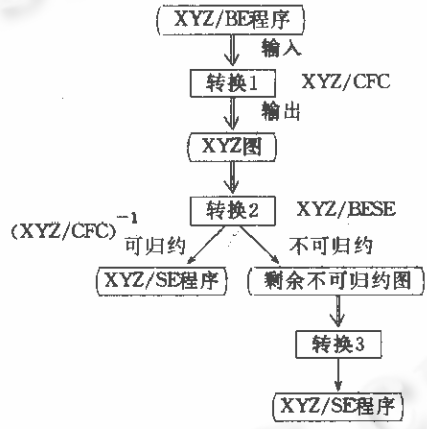


图2

容易想到,是否每一表示算法的 XYZ/BE 程序都能转换成 XYZ/SE 表示的结构化程序? XYZ 系统中 XYZ/BESE 系统即旨在提供一实现这种转换的工具.

2 结构化转换

本系统的基本思想是:对任给的一 XYZ/BE 程序先应用 XYZ/CFC[1] 自动作出其相应的 XYZ 图,然后用一组图形操作进行结构化归约. 如果整个图形都能进行这种归约,则得到一结构化的图,然后即可自动生成其 XYZ/SE 结构化程序. 如果这 XYZ 图中有不可归约的成分,就把剩下的非结构化部分用通常的结构化算法变成相应的 XYZ/SE 程序. 故整个系统的转换过程可用图形表示,如图 2.

定义. 一个 XYZ/SE 程序(简称 SEP)与一个 XYZ/BE 程序(BEP)等价. 即 SEP==BEP,如果 $SEP\wedge LB=START\Rightarrow\Diamond(R\wedge LB=STOP)$,当且仅当 $BEP\wedge LB=START\Rightarrow\Diamond(R\wedge LB=STOP)$. 其中 R 是任意的谓词.

任给一个 XYZ/BE 程序 $BEP=\%ALG[ce_1;ce_2;\ldots;ce_n]$,我们可以得到一个 XYZ 图(XYZ/CFC),产生 XYZ 图的算法:

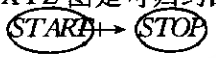
(1) 初始化图为空.

(2) 对程序中的每个条件元 ce_i，即 $LB=l_j \wedge P \Rightarrow \$\bigcirc Q \wedge \$\bigcirc LB=l_k$，检查标号节点 l_j 和 l_k 是否已在图中，如果 l_j 不在图中，将其加入进去，同样处理 l_k，然后做一条从 l_j 到 l_k 的有向边，边上标上条件 P，动作 Q.

(3) 从 1～n 做(2).

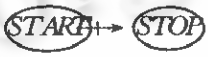
用此算法可得到任意 XYZ/BE 程序相应的 XYZ 图，其节点是条件元的标号，得到 XYZ 图是有限的，因为程序的条件元和条件元中的标号是有限的.

定义. 一个 XYZ/BE 程序的 XYZ 图是可归约的，如果它能被结构紧缩算法紧缩成



否则它是不可归约的(START 是开始标号而 STOP 是结束标号).

结构紧缩算法：

在 XYZ 图上做如下的变换直到 XYZ 图变成 

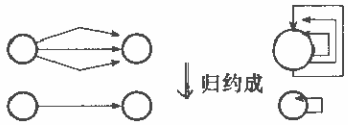
或不能再进行变换.

变换 1. 如果一个节点它仅有一条输入边和一条输出边→○→可归约成一条边→(去掉节点).

变换 2. 对自己到自己的边 

把自身边归约掉.

变换 3. 一组有同样起点和终点的边可归约到一条边



变换 4. 如果两个节点的度数都为 3，且结构如→○⇆○→，可归约成一条边→(去两个点).

对于任何可归约的 XYZ 图，总可以得到一个 XYZ/SE 程序，该程序与 XYZ 图所对应的 XYZ/BE 程序等价.

由可归约的 XYZ 图得 XYZ/SE 程序的算法：

做如下替换直到 XYZ 图变成一条有标记的边 $\xrightarrow{e}$，e 即是从 XYZ 图得到的 XYZ/SE 程序，这里引入一点标记.

对于一条边 $\xrightarrow{P}\ \xrightarrow{Q}$ P或Q均可以为空

$\Downarrow$

$\xrightarrow{e}$ e表示$P \Rightarrow \$\bigcirc Q$

如Q为空，则$P \Rightarrow$

如P为空，则 $\Rightarrow \$\bigcirc Q$

$cond(e)$表示 e 最左边⇒的左边的条件，$act(e)$表示最左边⇒的右边所有字符串.

有 2 个原算子：

0. *null* 表示空串

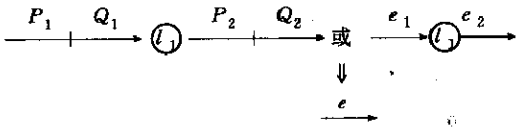
1. “:”表示字符串联接符

2. “if then else”

if p then f string 1 else string 2

$$=\begin{cases}\text{string 1 if } p \text{ 为真}\\ \text{string 2 if } p \text{ 为假}\end{cases}$$

替换 1.



e =if $act(e_1)\neq null$

then $\{e_1: \wedge\ \$\bigcirc LB=li\}$

else $\{e_1: \$\bigcirc LB=li\}$

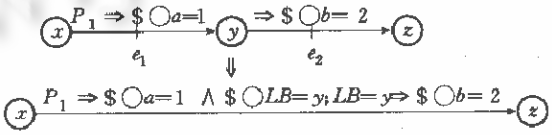
;if $con(e_2)\neq null$

then $\{LB=l \wedge ;e_2\}$

else $\{LB=l:e_2\}$

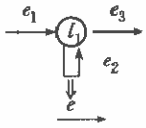
上面部分可简写成 $e_1 \wedge \$\bigcirc LB=l;LB=l \wedge e_2$

举例：



其中 $cond(e)=p_1;act(e)=\$\bigcirc a=1 \wedge \$\bigcirc LB=y;LB=y \Rightarrow \$\bigcirc b=2$

替换 2.



$e=e_1 \wedge \$\bigcirc LB=l_i;$

whiledo $[LB=l_1 \wedge e_2 \Rightarrow (\$\bigcirc LB=l[i] \mid \$\bigcirc LB=out[j]);$

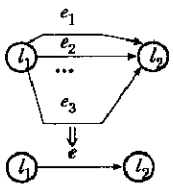
$LB=l[i] \Rightarrow act(e_2) \wedge \$\bigcirc LB=l_1;$

$]$

$LB=out[j] \wedge e_3$

其中 $l[i]$和 $out[j]$是标号产生器，由系统实现，$l[i]$为 $LBa-i$，$out[j]$为 $out-j$.

替换 3.



e=if $n=2$

then $\{cond(e_1) \vee cond(e_2) \Rightarrow \$\bigcirc LB=l[i];$

if $[LB=l[i] \wedge cond(e_1) \Rightarrow \$\bigcirc LB=l[i+1] \mid \$\bigcirc LB=l[i+2];$

$LB=l[i+1] \Rightarrow act(e_1) \wedge \$\bigcirc LB=out[j];$

$LB=l[i+2]\Rightarrow act(e_2)\wedge \$\bigcirc LB=out[j];$

]

$LB=out[j]\Rightarrow$

}

else if $n>2$

then $\{cond(e_1)\vee cond(e_2)\vee\ldots\vee cond(e_n)\Rightarrow \$\bigcirc LB=l[i];$

$case\ [LB=l[i]\wedge cond(e_1)\Rightarrow \$\bigcirc LB=l[i+1]$

$|cond(e_2)\Rightarrow \$\bigcirc LB=l[i+2]$

...

$|cond(e_n)\Rightarrow \$\bigcirc LB=l[i+n];$

$LB=l[i+1]\Rightarrow act(e_1)\wedge \$\bigcirc LB=out[j];$

$LB=l[i+2]\Rightarrow act(e_2)\wedge \$\bigcirc LB=out[j];$

...

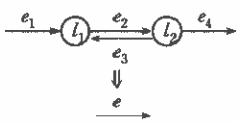
$LB=l[i+n]\Rightarrow act(e_n)\wedge \$\bigcirc LB=out[j];$

]

$LB=out[j]\Rightarrow$

}

替换 4.



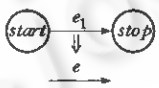
$e=e_1\wedge \$\bigcirc LB=l_1;$

$LB=l_1\wedge e_2\Rightarrow \$\bigcirc LB=l_2;$

$LB=l_2\wedge e_3\Rightarrow(\$\bigcirc LB=l_1|\$\bigcirc LB=out[j]);$

$LB=out[j]\wedge e_4$

替换 5.



$e=\%ALG[LB=START\wedge e_1\wedge \$\bigcirc LB=STOP;]$

一个固有的 XYZ 图有如下结构：

(a)p 有一个入口边和一个出口边

(b)对于每个图中的节点，有一条从入口边到出口边的路经过它.

显然，可归约的 XYZ 图是固有的.

对于任何一个不可归约的固有的 XYZ 图，也能够得到一个等价的 XYZ/SE 程序.

算法：

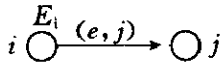
(a)用前一个算法做替换直到不可归约的 XYZ 图到不能变换为止.

(b)对于剩下的不可归约的部分，首先加一条 λ 边在 START 上，加一条出

边在STOP上,然后任意把图中的节点编号.START为1号,STOP为0边上的号为它到达的节点的号,这样边上有标识对(e,i).

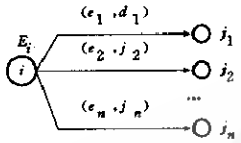
(c)引入一个新的逻辑变量 *count*(它不在原来的 *XYZ/BE* 中出现),给每个节点标记 $E_i, i=1,2,\ldots,n$.

1. 对仅有一个出边的节点



$$E_i = e \wedge \$\bigcirc count = j$$

2. 对于有多于一条出边的节点 i



$$E_i \Rightarrow \$\bigcirc LB = l[i];$$

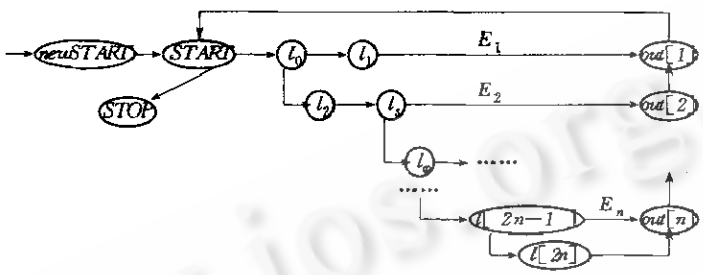
$$case[LB = l[i] \wedge cond(e_1) \Rightarrow \$\bigcirc LB = l[i+1]$$
$$| cond(e_2) \Rightarrow \$\bigcirc LB = l[i+2]$$
$$\ldots$$
$$| cond(e_n) \Rightarrow \$\bigcirc LB = l[i+n]$$
$$LB = l[i+1] \Rightarrow act(e_1) \wedge \$\bigcirc coust = j_1 \wedge \$\bigcirc LB = out[j];$$
$$LB = l[i+2] \Rightarrow act(e_2) \wedge \$\bigcirc coust = j_2 \wedge \$\bigcirc LB = out[j];$$
$$\ldots$$
$$LB = l[i+1] \Rightarrow act(e_n) \wedge \$\bigcirc coust = j_n \wedge \$\bigcirc LB = out[j];]$$
$$LB = out[j] \Rightarrow$$

这样得到一个 while 结构的程序,程序体是一些测试 coust 值的条件语句每个条件语句的 then 部分是

$$\%ALG[LB = newSTART \Rightarrow \bigcirc coust = 1 \wedge \bigcirc LB = START;$$
$$LB = START \wedge (coust >= 0) \Rightarrow (\$\bigcirc LB = l[o] | \$\bigcirc LB = STOP),$$
$$if[LB = l[o] \wedge count = 1 \Rightarrow \$\bigcirc LB = l[1] | \$\bigcirc LB = l[2];$$
$$LB = l[1] \wedge E1 \wedge \$\bigcirc LB = out[1];$$
$$if[LB = l[2] \wedge coust = 2 \Rightarrow \$\bigcirc LB = l[3] | \$\bigcirc LB = l[4];$$
$$LB = l[3] \wedge E2 \wedge \$\bigcirc LB = out2;$$
$$if[LB = l[4] \wedge coust = 3 \Rightarrow \$\bigcirc LB = l[5] | \$\bigcirc LB = l[6];$$
$$\ldots$$
$$if[LB = l[2n-2] \wedge coust = n \Rightarrow \$\bigcirc LB = l[2n-1] | \$\bigcirc LB = l[2]$$
$$LB = l[2n-1] \wedge En \wedge \$\bigcirc LB = out[n];$$
$$LB = l[2n] \Rightarrow \$\bigcirc LB = out[n];$$
$$]$$
$$LB = out[n] \Rightarrow \$\bigcirc LB = out[n-1];$$
$$\ldots$$
$$]$$

```
        LB=out[2]⇒ $ ○LB=out[1];
        ]
    LB=out[1]⇒ $ ○LB=start;
    ]
]
```

该 XYZ/SE 程序对应的 XYZ 图为



致谢　本文是作者在中国科学院软件所唐稚松教授指导下完成的硕士论文的一个组成部分. 因匆匆赴美，承严海林等同学帮助在文章及系统的符号等方面进行了一些整理与修改，在此一并表示感谢.

参考文献

1　龚洁，唐若鹰，王霄等. XYZ/CFC 与 XYZ/PAD：图形——文本程序设计环境. 软件学报，1994，5(8)：37～46.

2　龚洁. XYZ/CFC 的实现[硕士论文]. 中国科学院软件所，1989.

3　Xie H *et al*. A structured temporal logic language：XYZ/SE. J of Comput. Sci & Technol，1991，6(1).

4　Tang C S. XYZ：a CASE environment based on temporal logic and conforming to multi—ways of programming. (revised version) Tech. Rep. No. ES. CAS—XYZ—91—1. Enst. of Softw. Acad. Sini.，1991.

5　Linger R C *et al*. Structured programming，theory and practice. Addison—Wesly，1979. 112～126.

AN INTERACTIVE GRAPHIC TOOL TO TRANSFORM A PROGRAM INTO A STRUCTURED FORM

Gong Jie

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

Abstract　This tool is based on a graphic language called XYZ/CFC which is the graphic representation of a temporal logic language XYZ/E. By means of this tool，a XYZ/E program of unstructured form can be transformed into structured form. XYZ/E is suitable to play the role of an intermediate language. As the comequence，this tool can also be used to transform a program of any conventional language into its structured form.

Key words　Structured higher level language，structuring program transformation，graphic design tool，temporal logic language.