第5卷第8期
1994年8月

软 件 学 报
JOURNAL OF SOFTWARE

Vol. 5, No. 8
Aug. 1994

XYZ/CFC与XYZ/PAD：图形—文本程序设计环境*

龚 洁　唐若鹰　王 霄　唐稚松

（中国科学院软件研究所，北京 100080）

摘要 XYZ系统是一个CASE工具系统.它的核心是一个时序逻辑语言XYZ/E.XYZ/E有一基本的表示状态转换的低级形式XYZ/BE(或用于表示并发的XYZ/CE)及一个结构化的高级形式XYZ/SE.它们均有其相应的图形表示.XYZ/CFC与XYZ/PAD是分别以XYZ/BE(或XYZ/CE)及XYZ/SE用逐步求精方法进行程序设计的交互式的图形环境.每步均可由图形程序自动生成时序逻辑形式的程序.

关键词 时序逻辑，操作语义，语义一致性.

XYZ系统中的核心是一时序逻辑语言XYZ/E，它的一个重要特征是可以以统一的程序框架表示抽象描述及可执行的过程性算法程序.因此，XYZ系统能支持逐步求精的程序设计方法，从完全抽象描述，抽象描述与过程性算法的不同程度的掺合，一直到完全由可有效执行的算法过程组成的程序；对于每一步得到的描述，即可用系统中提供的验证工具(XYZ/VERI)验证上下两步描述的语义一致性，也可用系统中速成原型工具(XYZ/PROT，XYZ/RULE)执行每步描述，用求得的值比较前后步语义一致性.

近年来，软件工程研究表明，以图形表示程序可充分发挥二维显示终端的特色，增强直观性.而XYZ/E语言又正好便于表示图形的语义.因此，系统中提供一种图形语言XYZ图，它与XYZ/E同构，可以说是XYZ/E时序逻辑语言的图形表示.用它同样可表示出逐步求精过程.XYZ/CFC即是支撑以逐步求精方式设计XYZ图程序的交互式图形环境.而且，在每一步图形设计出以后，即可自动生成相应的XYZ/E文本程序.根据所对应的XYZ/E文本，亦即可验证或以速成原形方法检验对图形进行分解时上下两步语义一致性.

国际上著名的图形程序设计系统Statemate[7]及SDL与XYZ/CFC具有相近的特色.不过，XYZ/CFC更侧重图形与时序逻辑语言XYZ/BE，XYZ/CE及XYZ/SE各成分之间的对应关系，以便于由图形自动生成表示其语义的时序逻辑程序.表示的内容更接近通常程序设计的习惯，且十分简单易学.

* 本文1992—03—06收到，1992—05—30定稿

作者**龚洁**，女，30岁，助研，主要研究领域为软件工具与环境.**唐若鹰**，女，25岁，助研，主要研究领域为软件工具与环境.**王霄**，女，25岁，助研，主要研究领域为软件工具与环境.**唐稚松**，69岁，中国科学院院士，研究员，主要研究领域为计算机科学与软件工程.

本文通讯联系人：唐稚松，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

XYZ图与XYZ/E语言也经常遇到一种意见,即它不具有一般高级语言的控制结构,是一较低级的语言形式.这种形式在表示Petri网等状态转换图形方面有其方便之处.但也有非结构化的缺点.为此,在XYZ系统中提供一结构化高级语言形式的时序逻辑语言XYZ/SE,它只是对XYZ/E稍加扩充(或限制)而成.对这种时序逻辑语言,也有相应的图形表示XYZ/PAD.它是在日本人提出的PAD图之上扩充表示抽象描述及并发机制而成.用这种图形也同样可表示上述逐步求精过程,并验证或检验上下两步的语义一致性.同时,在每步图形设计之后,亦可自动生成其XYZ/E程序.

上述两环境下均可与XYZ系统中提供的分布式进程设计环境XYZ/DPD或XYZ/DFD结合使用.比如,应用此系统设计按SA方法开发的实时信息系统.我们已着手在高炉过程控制软件设计方面进行了一些探索.

1 XYZ图

XYZ图是一表示时序逻辑语言XYZ/E的图形语言,它展示了XYZ/E程序的控制结构,而XYZ/E则给出了这图形语言的操作语义.XYZ图不同于传统的控制流图,它是状态图(STATE CHART)与PETRI网的结合体,能表示抽象描述及过程性算法且具有层次结构,可进行逐步分解,而且它还可表示并发,与PETRI网相应,每个位置(PLACE)对应于一个标号(状态),而每个传递(TRANSITION)对应通讯进程的"握手"(HANDSHAKING).

XYZ图是一个有向图,由节点及连接节点的边组成.

节点有如下几类:

(1)标号节点:圈 ○

表示程序状态每个节点中有一个标号和一个进程名.在同一进程中的不同标号节点其标号各不相同.当只有一个进程时,进程名省略不写.这种节点输入边及输出边个数任意.

(2)等待节点:重叠 ▭

用于并发进程的同步中,输入边及输出边的个数至少为2.

(3)定义或约定部分节点:框 □

有向边有如下几类:

(1)定义边 ⇨ 用作定义或约定部分节点的输出边

(2)无条件转移边 动作→ 用作输出边,仅有动作无条件部分,动作可为空

(3)有条件转移边 条件/动作→ 为主要的转移边,条件或动作均可为空

动作部分随着时序算子的不同又可有不同的形式

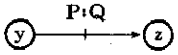
——→ 实线 表示$O(下一时刻,NEXTTIME)

- - -→ 虚线 表示◇(终于,EVENTUALLY)

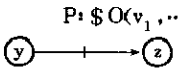
 表示$U

© 中国科学院软件研究所 http://www.jos.org.cn

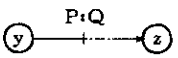
条件元分以下几种形式:



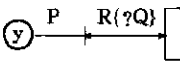
$lb=y \wedge P \Rightarrow \$O(Q \wedge lb=z)$



$lb=y \wedge P \Rightarrow \$O(v_1,\cdots,v_k)=(e_1,\cdots,e_k) \wedge \$Olb=z$



$lb=y \wedge P \Rightarrow \Diamond(Q \wedge lb=z)$



$lb=y \wedge P \Rightarrow R\ \$U(Q \wedge \$Olb=z)$

对于一组入口标号相同的条件元:

$lb=y \wedge P_1 \Rightarrow @_1(Q_1 \wedge lb=z_1)$

…

$lb=y \wedge Pk \Rightarrow @_k(Q_k \wedge lb=z_k)$

其表示成如图 1.

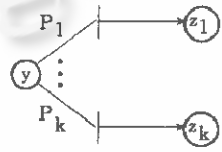


图1

此处$@_i$表示算子"$O"或"◇".当$@_i$为◇时,则将相应于它的实线换成虚线.

对于以下各有特点的条件元

$lb_i=y \wedge P \Rightarrow @(Q_1 \wedge lb_{j_1}=z_1 \wedge Q_2 \wedge lb_{j_2}=z2)$;

$lb_i=y \wedge P \Rightarrow @_1(Q_1 \wedge lb_{j_1}=z_1)\ \$ \wedge @_2(Q_2 \wedge lb_{j_2}=z_2)$;

$lb_i=y \wedge P \Rightarrow @_1(Q_1 \wedge lb_{j_1}=z_1)\ \$V @_2(Q_2 \wedge lb_{j_2}=z_2)$;

$lb_i=y \wedge P \Rightarrow @_1(Q_1 \wedge lb_{j_1}=z_1)\ \$V' @_2(Q_2 \wedge lb_{j_2}=z_2)$;

其表示分别为如图 2.

每个条件元对应一个有向子图,因此 XYZ/E 单元(即条件元的合取)与一个有向图对应,完整的图形形式为如图 3.

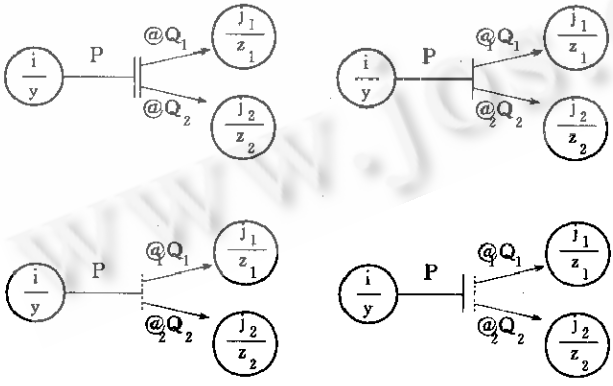


图2

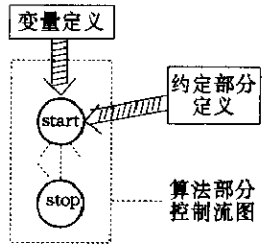


图3

以下给出用 XYZ 图表示 XYZ/E 程序的例子.

例 1:阶乘 M! 的 XYZ/E 程序

fact:%alg[$lb=fact\_start \Rightarrow \$Oz=1 \wedge \$Oj=1 \wedge \$Olb=l1$;

$lb=l_1 \wedge j<m+1 \Rightarrow \$Oz=z*j \wedge Oj=j+1 \wedge Olb=l1$;

© 中国科学院软件研究所 http://www.jos.org.cn

lb=l_1 ∧ j>=m+1⇒ $Of=z ∧ $Olb=fact_stop]

where %inp(m:I) ∧ outp(f:I) ∧ %var(z:I;j:I)

其 XYZ 图如图 4:

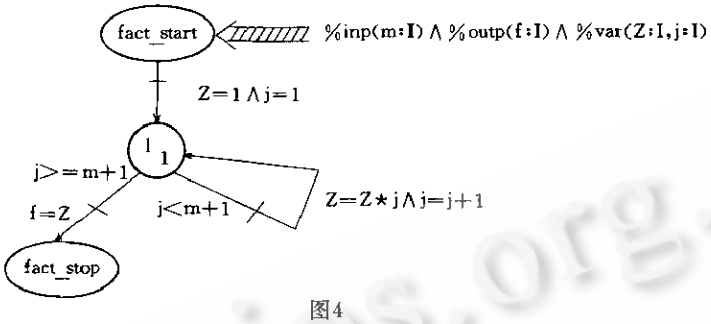


图4

例 2:求阶乘和 $\sum_{i=1}^{K} n!$ 的 XYZ/E 程序

sum:%alg[lb=sum_start⇒ $Oi=0 ∧ $Os=0 ∧ $Olb=$l_1$;

lb=l_1 ∧ i=k+1⇒ $Olb=sum_stop;

lb=l_1 ∧ i≠k+1⇒ $Olb=$l_2$;

lb=l_2⇒◇(f=i! ∧ lb=l_3);

lb=l_3⇒ $Os=s+f ∧ $Oi=i+1 ∧ $Of=1 ∧ Olb=$l_1$]

这里我们省略了变量及阶乘运算"!"的定义部分.

SUM 是个抽象程序,也是 XYZ/E 程序,它反映了逐步求精过程的一个中间阶段,下面对 SUM 继续求精得:

其 XYZ 图为图 5.

将虚线分解替换得如图 6.

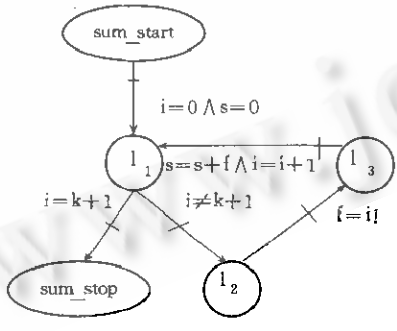


图5

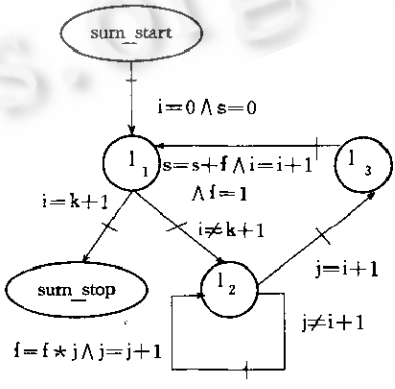


图6

例 3:用 XYZ/E 写通讯进程间的通讯

S:发送者

© 中国科学院软件研究所 http://www.jos.org.cn

R:接收者

进程 S 与 R 由通道 CH(s—r)连接,每方有缓冲区 s_buf,r_buf,满足公理[][$O_buf=s—buf],由硬件实现.

引入谓词

SEND(s,r):从进程 S 向进程 R 送数据

REC(r,s):从进程 R 向进程 S 接收数据

通讯原语 P_s⇒CH(s—r)! x 及 P_r⇒CH(s—r)? y 可用 XYZ/E 程序表示为:

s:%alg[…

lbs=ls_1∧P_s⇒$OSEND(s,r)∧$Os—buf=x∧$Olbs=$ls_2$;

lbs=ls_2∧SEND(s,r)⇒(SEND(s,r))$U(REC(r,s)∧$Olbs=ls_3);

lbs=ls_3⇒lbs=$ls_3$$U(THRU∧$Olbs=NEXTs;

…]

r:%alg[…

lbr=lr_1∧P_r⇒$OREC(r,s)∧$Olbr=lr_2;

lbr=lr_2∧REC(r,s)⇒REC(cr,s)$U(SEND(s,r)∧$Olbr=lr_3);

lbr=lr_3⇒$Oy=r_buf∧$O(THRU)∧$Olbr=NEXTr;

…]

其中 THRU 是一谓词,只要接收方完成数据接收后,它便为真.这里时序逻辑公式"M$U(N∧$Olb=y)"定义为:(M$UN)∧(N→$Olb=y).详细的讨论可查阅文献[8].

通讯进程的 XYZ 图为图 7.

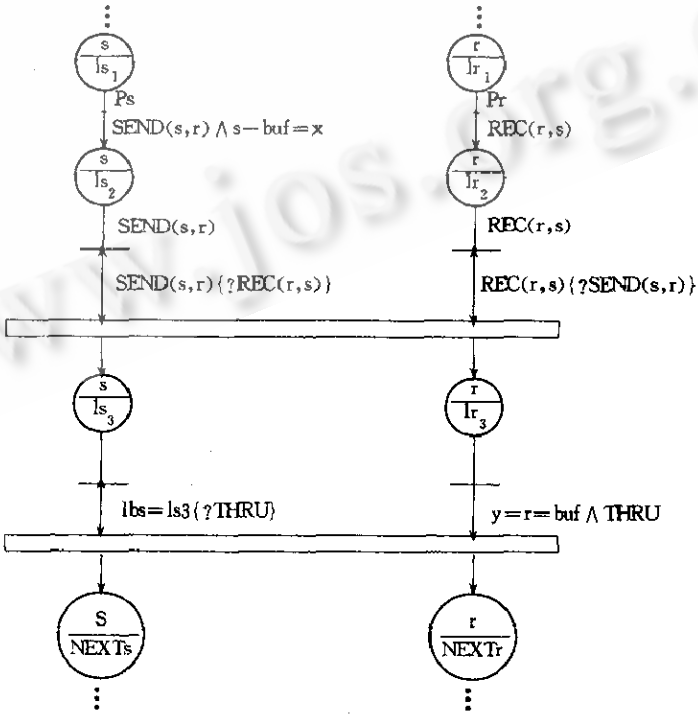


图7

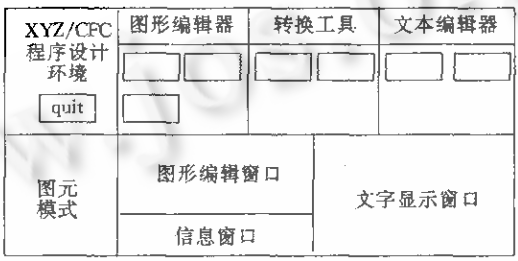
© 中国科学院软件研究所 http://www.jos.org.cn

由上可以看出,由图表示比文本直观,自然,可读性更强,修改更方便.

2 XYZ图形设计环境XYZ/CFC

XYZ/CFC图形设计环境包括3个部分.(1)图形编辑器:用于编程修改XYZ图图形语言程序.(2)文本编辑器:用于编程修改XYZ/E语言程序,功能与UNIX系统的VI完全一致.(3)自动转换工具:a.从XYZ/E程序到XYZ图的自动转换.b.从XYZ图到对应的XYZ/E程序图的转换.

整个系统由菜单驱动,具有友好的界面,有Message窗口对每一步进行文字提示,告诉用户下一步该怎么做.系统启动时出现一主窗口,窗口由以下部分组成:



[quit]退出整个系统.

图形编辑器包含的功能有:

[LOAD][SAVE][REDRAW][DESIGN][MOVE][EXIT][FILE][ERASE][DECOMPOSE]功能描述:

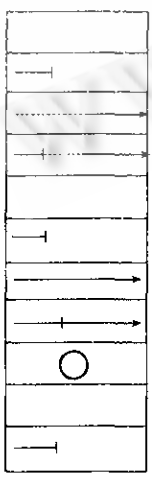
[LOAD]将一个图形文件取到图形编辑器中,对应XYZ图将出现在图形编辑窗口上.

[SAVE]将当前的XYZ图存入一个图形文件.

[REDRAW]在图形窗口上重画当前的XYZ图.

[DESIGN]利用图元模式来设计流图,在设计过程中进行语法检查,保证用户设计出的图形是一个XYZ图.

用户可通用的图形元素包括:



ENTRY1:表示无条件的EVENTUALLY◇,曲线边

ENTRY2:表示有条件的EVENTUALLY◇,曲线边

ENTRY3:表示无条件的EVENTUALLY◇,直线边

ENTRY4:表示有条件的EVENTUALLY◇,直线边

ENTRY5:表示无条件的VEXT TIME $O,曲线边

ENTRY6:表示有条件的VEXT TIME $O,曲线边

ENTRY7:表示无条件的NEXT TIME $O,直线边

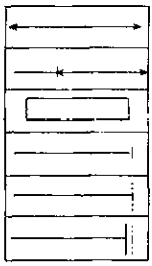
ENTRY8:表示有条件的NEXT TIME $O,直线边

ENTRY9:表示标号节点

ENTRY10:表示无条件的UNTIL $U,曲线边

ENTRY11:表示有条件的UNTIL $U,曲线边

© 中国科学院软件研究所 http://www.jos.org.cn



ENTRY12:表示无条件的UNTIL $U,直线边

ENTRY13:表示有条件的UNTIL $U,直线边

ENTRY14:表示等待节点

ENTRY15:表示条件，其后的动作之间的关系是合取 $^

ENTRY16:表示条件，其后的动作之间的关系是析取 $v

ENTRY17:表示条件，其后的动作之间的关系是异或 $v'

[MOVE]在窗口上移动当前的 XYZ 图,选择 MOVE 后,会出现

△与▲功能一样，只是▲更快

[EXIT] 退出图形编辑器,当前与图有关的一切数据结构被初始化.

[FILL] 用于文字填写,在标号节点中填写标号或进程名,在边上的条件和动作中填入条件与动作.

[ERASE]用于删除图,可删边,点,子图或全部图.

[DECOMPOSE] 用于对条件元进行分解,对一个子图进行替换,当用户选择好替换的部分后,用户做分解的窗口出现了,见下图:

此窗口中用户可以编辑图形，如按SUBSTITUTE，则将此图形替换到所选的子图，退出分解状态.
如按FINISH，则不替换，直接退出分解状态.

文本编辑器的功能包括:

[LOAD]装载一个文件且在文字显示窗口上显示.

[SAVE]存文本文件于 OS 的文件系统中.

[MOVE]在文本显示窗口上移动显示信息.

[EXIT]退出文本编辑器,各种有关的数据结构被初始化.

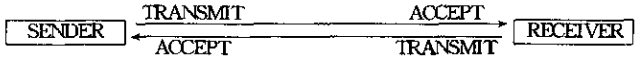
[MODIFY]功能与 UNIX 中的 VI 完全一致.

转换工具器的功能包括:

[CHART TO TEXT]由 XYZ 图转换到与其对应的 XYZ/E 程序.

[TEXT TO CHART]由 XYZ/E 程序得到相应的 XYZ 图,其中利用了一些图的布局的算法,这里就省略了.

例如:AB PROTOCAL 的 SENDER 方,从其抽象描述得到可执行的算法



NEXT:指向下一个要被送的信息或已经被送未被确认的信息的序号.

LAST_CONFIRMED:指向已经被确认的所有序号中的最大序号. 初始时,LAST－CONFIRMED 设为－1,NEXT 为 0.

根据上面所述,我们可用 XYZ CHART 写出 SENDER 的抽象描述,如图 8.

～ACCEPT(ACK(J))　　没收到 J 的确认

© 中国科学院软件研究所 http://www.jos.org.cn

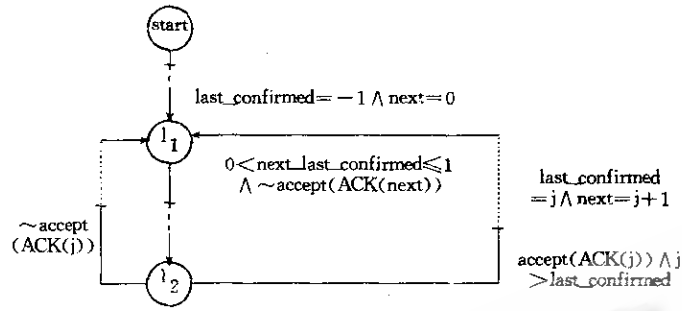


图8

这个谓词的实现与时间有关，如超时其时间片就表示没收到，重发，而不能无限等下去.

START(I)=T(RUE)　　开始对I计时

CANCEL(I)=T(RUE)　　对I关闭时间

TIMEOUT(I)=T(RUE)　　I的时间片到

经过这步求精，替换(L_1)到(L_2)之间的部分，XYZ图变成如图9.

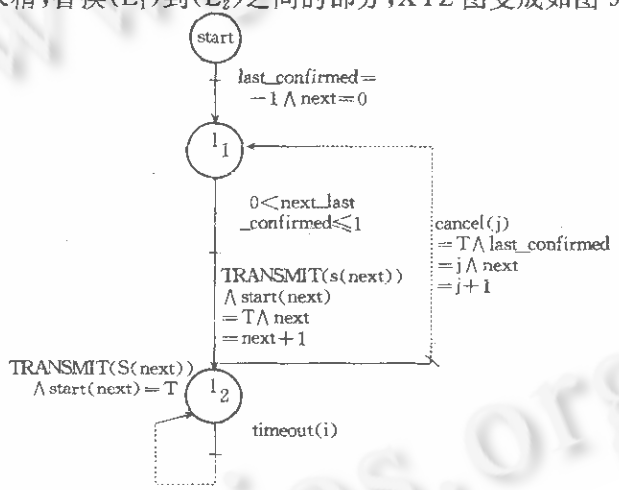


图9

分解和替换在程序从抽象描述变成可执行算法时起到很重要的作用，因为这是一个逐步求精的过程，是具有层次结构的，低层实现高层，低层的性质能推出高层的性质. 上下层语义要求一致. 在本系统中，我们提供了图形分解及替换的功能，但没提供语义检查以确保分解与替换能维持语义一致性. 显然，这件事不能全自动实现. 但在XYZ系统中，提供了两种办法来检验语义一致性：一是用时序逻辑的验证系统XYZ/VERI来检验图形分解前后的相应的XYZ/E程序. 另一种办法是将图形分解前后的XYZ/E程序作为快速原型来执行，用一组数据去测试执行结果是否一致.

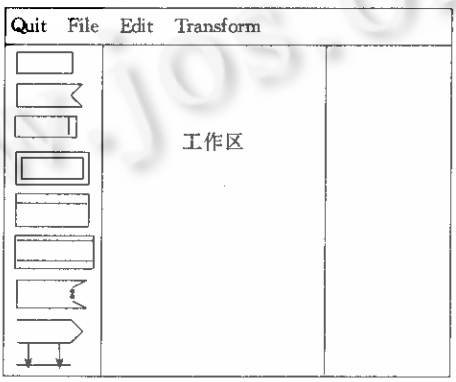
XYZ/PAD是日本PAD[6]的扩充. 它增加了表示抽象描述及并发性的功能. 它比XYZ图具有更为高级的结构形式. XYZ/PAD则是这种图形的一连结构化编辑器. 由于其直观性及排错功能，使用户更为方便.

XYZ−PAD是一树形结构图，由以下图元组合而成：

© 中国科学院软件研究所 http://www.jos.org.cn

(1)顺序框：
(2)选择框：
(3)重复框： 或 (后判断循环)
(4)定义框：
(5)子程序处理框：
(6)进程处理框：
(7)case语句处理框：
(8)并发语句处理框：

XYZ—PAD程序设计环境的窗口界面布局如：



它的功能描述如下：

菜单 FILE 包括那些以 PAD 图文件为对象或与存取文件有关的操作：

[NEW]　清除工作区的内容,建立一个新的 PAD 文件.

[OPEN]　打开一个已经存在的 PAD 文件并装入工作区.

[SAVE]　把工作区内的 PAD 图存入指定的文件.

[PRINT]　通过打印设备输出 PAD 图.

菜单 EDIT 包括那些对 PAD 图进行操作的一切操作：

[DELETE]　删除 PAD 图中的某一部分.

[MODIFY]　对 PAD 图中某个框的文字内容以及补充说明进行修改.

[DECOMPOSE]　对图中定义框进行分解.

[UP]　从工作区中子 PAD 图返回到它的上层.

菜单 TRANSFORM 包括由 PAD 图到 XYZ/SE 的自动转换以及由 PAD 图到 C 语言的自动转换.

[TO XYZ/SE]　PAD 到 XYZ/SE 的自动转换.

[TO C]　PAD 到 C 程序的自动转换.

[C TO PAD]分解在逐步求精的过程中起一个非常重要的作用.这种求精过程是层次性的.低层实现高层.它要求上下层语义一致性.在 XYZ/PAD 中,我们提供了图形求精的功能,但没提供语义检查来确保某个分解是确保语义的.这个工作是不能自动生成的.然而用户可以首先通过自动转换工具把 XYZ—PAD 转换成类 C 程序.由于类 C 程序是标准 C

© 中国科学院软件研究所 http://www.jos.org.cn

语言+抽象的SPECIFICATION语句，所以我们可以根据上下层的类C程序来检查其语义一致性，从而可知相应的XYZ—PAD的语义一致性.

在XYZ中还包含一工具自动将一非结构化的XYZ/BE程序变成一结构化的XYZ/SE程序.这工具名为XYZ/B—SE，它是XYZ/CFC为工具实现的，我们将在另文中介绍.

3 结 论

在本文中，我们讨论了图形语言——XYZ chart和XYZ PAD的背景、特点及应用.同时我们描述了XYZ/CFC与XYZ/PAD程序设计环境的功能. XYZ/CFC有一很有意义的应用，即非结构化的XYZ/E程序转换成结构化的XYZ/SE程序，详见另文.

参考文献

1 Tang C S. XYZ: A case environment based on temporal logic and to unify multi—ways of programming. 中国科学院软件所技术情报 IS—CAS—XYZ—91—1，1991.

2 冯玉琳，林惠民，唐稚松. 时序逻辑的证明系统. 计算机研究与发展，1985，22(10).

3 Miao Xu. A methodology and environment for stepwise refinement——according to design decisions. M. S. Thesis, Inst. of Software, 1989.

4 Gong Jie. The implementation of XYZ/CFC. M. S. Thesis, Institute of Software, 1989.

5 Linger R C, Mills H L, Witt B I. Structured programming theory: and practice. Addison—Wesley, 1979: 12—126.

6 冯玉琳，赵保华. 软件工程——方法、工具和实践. 北京：中国科学技术大学出版社.

7 Harel D *et al*. Stalemate: A working environment for the development of complex reactive system. Proc. of Intl. Conf. on Software Engineering, 1988.

8 Tang C S. A temporal logic language oriented toward software engineering. 中国科学院软件所技术报告，1993.

XYZ/CFC AND XYZ/PAD: THE GRAPHIC—TEXTUAL PROGRAM DESIGN ENVIRONMENTS

Gong Jie, Tang Ruoying, Wang Xiao and Tang Zhisong

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

Abstract XYZ system is a CASE tool system. Its kernel is a temporal logic language XYZ/E which has a basic lower levels state—transition form XYZ/BE (or XYZ/CE for concurrency) and a structured higher level form XYZ/SE. Each of them has a corresponding graphic representation. XYZ/CFC and XYZ/PAD are two interactive graphic environments to support programming with stepwise decomposition methodology by means of XYZ/BE (or XYZ/CE) and XYZ/SE respectively. The Temporal logic form programs can be generated automatically from the graphic programs at each step.

Key words Temporal logic, operating semantics, consistency of semantics.

© 中国科学院软件研究所 http://www.jos.org.cn