程序性质的描述及证明*)

官荷卿　郭　亮
(中国科学院软件研究所计算机科学开放研究实验室　北京100080)

Description & Verification of a Program's Property

GUAN He-Qing　GUO Liang
(Lab of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing 100080)

Abstract In this paper, we specify and implement the Dekker algorithm in XYZ/AE and XYZ/EE respectively, the semantic consistency between the specification and its implementation is also proved under the framework of temporal logic.

Keywords Temporal logic language XYZ/E, Dekker algorithm, Specification, Prove, Liveness, Safety

XYZ/E[1,2]是一种基于Manna-Pnueli线性时序逻辑的线性时序逻辑语言(LTLL),其主要特征为它在统一的时序逻辑框架下既能表示程序的静态规范(XYZ/AE)也能表示可执行代码(XYZ/EE),因此程序规范和程序可执行代码的语义一致性也就得以在时序逻辑框架下验证。

对于顺序程序,XYZ系统提供了一套基于Hoare逻辑规则的验证工具XYZ/VERI。此工具通过读取程序及其前后断言和所有循环语句的不变量,用机械的方法产生一一阶逻辑表达式。若能证明此表达式为真,则我们就可得到此程序的部分正确性。整个方法存在的问题为现在没有行之有效的方法自动找出循环不变量来。

对于并行程序以及顺序程序的可终止性(或完全正确性),现在还没有特别有效的方法给予证明。但由于XYZ/E本身是一种时序逻辑语言,因此我们可以借用时序逻辑语言对一些程序性质的证明方法来证明XYZ/E语言程序的性质。

本文中,我们采用XYZ/EE语言实现了Dekker算法[3],用XYZ/AE语言刻画其安全性和活性[4],最后用与时序逻辑程序[5]相对应的XYZ/E程序证明规则证明我们实现的算法与刻画出的性质之间的语义一致性。

1　XYZ/E简介

时序逻辑语言XYZ/E是一个一阶线性多类型(many-sorts)逻辑系统,它既是一个逻辑系统,又是一个程序设计语言,其数据类型包括整型、字符型及布尔型等。XYZ/E语言既可以作为规范语言表示系统的静态规范(静态语义),又可以作为系统刻画语言描述系统的动态行为(动态语义),为此XYZ/E被称为世界上第一个可执行的时序逻辑语言[6],下面我们将介绍一些本文中用到的XYZ/E语言的符号表示方法。

1.1　状态转换等式

状态转换等式形式如式(1)所示,用于将状态转换机制表示在直言式逻辑中。此处$O为下一时刻算子,$v$表示一时序变量,$e$表示与$v$具有相同数据类型的表达式,其中不允许出现任何时序算子。整个式子的含义是:v在下一时刻的值等于表达式e在本时刻的值。事实上这就表示了常见高级语言中的赋值语句。若变量v为特殊的控制变量"LB",它恒取程序当前的执行标号为其值,其形式如式(2)所示,含义即"下一时刻程序执行标号为y"。这即表示了常见高级语言中的跳转语句。

$$\$Ov=e \tag{1}$$

$$\$OLB=y \tag{2}$$

1.2　条件元

XYZ/E语言中最基本的元素是条件元(Conditional Element)。条件元共有三种形式,式(3)所示条件元直接定义了相邻状态之间的转换关系,因此全部由此种形式条件元组成的程序可以执行。其它两种条件元分别表示程序的抽象规范和产生式规则。其中,标号y称为条件元的定义标号,标号z称为条件元的转出标号。

$$LB=y\wedge R\Rightarrow \$O(v_1,v_2,\cdots,v_n)=(e_1,e_2,\cdots,e_n)\wedge \$OLB=z \tag{3}$$

1.3　单元

XYZ/E中表示算法的构件为单元(Unit),形式如式(4)所示。其中$A_1,A_2,\cdots,A_n$是条件元,符号";"等同于逻辑联结词合取。单元中条件元可以是表示状态转换的可执行条件元,也可以是表示程序规范的条件元,因此可以描述程序开发的不同阶段。

$$\Box[A_1;A_2;\cdots;A_n] \tag{4}$$

对于单元及单元中的所有条件元,存在如下四点假设:

1)关于时序变量的框架假设:对于形如式(3)所示的可执行条件元中没有显式给定下一时刻值的变元,都假定其下一时刻值等于当前时刻值。

2)关于标号的正则性假设:每个单元存在唯一入口标号START;单元的转出标号为STOP(单元表示进程的执行体)或STOP(单元表示过程的执行体);单元中每个条件元的转出标号都对应单元中某个其它条件元的定义标号。

3)关于条件的完整性假设:单元中具有相同定义标号的条件元称为同源的,一单元中所有条件元按其同源性可分化为不相交的同源集。一同源集称为完整的,当且仅当此同源集中各条件元的条件的或等价于逻辑真。而单元的关于条件的

*)本文研究得到国家自然科学基金YCK15028资助。官荷卿　硕士生,主要研究领域为软件工程。郭　亮　博士生,主要研究领域为软件工程。

完整性假设即为单元中各同源集都是完整的。

4)关于条件的互斥性假设:单元的同源集内任意两个条件的交都等价于逻辑假。

1.4 循环语句和分情形语句

XYZ/E 中除了存在对应于表示高级语言的转出语句意义的如式(3)所示条件元外,还存在循环、分情形两种语句分别与高级语言常见的循环和分支两种控制结构相对应。

1)循环语句 其完整形式是:

```
*[LB=EntryLabel ∧ R⇒($OLB=y | $OLB=EXIT);
  LB=y{|S|} $OLB=EntryLabel
]
```

其中{|S|}表示一条语句。

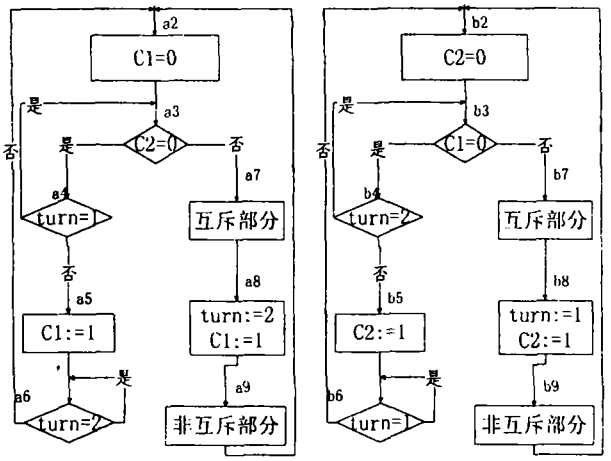
2)分情形语句 其完整形式是:

```
?[LB=EntryLabel ∧ R1⇒$OLB=Label1
                |R2⇒$OLB=Label2
                  …… ……
                |Rk⇒$OLB=Labelk;
  LB=Label1{|S1|}$OLB=EXIT;
                  …… ……
  LB=Labelk{|Sk|}$OLB=EXIT ]
```

2 Dekker 算法及其描述

Dekker 算法为两个进程 P_1、P_2通过共享三个变量 C_1、C_2 和 turn 实现互斥访问临界区。进程 P_1和 P_2的流程图如下:(算法的具体描述见参考文[3])

其中,C_1、C_2、turn 的初始值都为1。



进程 P_1 进程 P_2

令 $\pi \equiv initial \wedge C_1=1 \wedge C_2=1 \wedge turn=1$

cobegin $P_1 || P_2$ coend

则用时序逻辑我们可以刻画出程序 π 具有如下两个性质。

(1)安全性

即 $at\alpha_2 \wedge at\beta_2 \wedge C_1=1 \wedge C_2=1 \wedge turn=1 \rightarrow \Box\neg(at\alpha_7 \wedge at\beta_7)$

此性质表明进程 P_1和 P_2不能同时进入互斥部分。

(2)活性

即 $at\alpha_2 \wedge at\beta_2 \rightarrow \Diamond(at\alpha_7 \wedge at\beta_7)$

此性质表明在公平性的基础上,进程 P_1和 P_2从入口点 α_2 和 β_2开始,最终一定进入互斥部分(假定 P_1和 P_2在进入临界区后,有限时间内都会退出并释放临界区)。

3 使用 XYZ/E 描述和实现 Dekker 算法

下面我们用 XYZ/EE 语言写出 Dekker 算法,用 XYZ/AE 语言描述其规范。为简单起见,我们并不写出整个对应的 XYZ/EE 源程序,而只写出进程 P_1和 P_2对应的 XYZ/EE 算法,以及刻画出程序 $P=||[P_1,P_2]$的安全性和活性。

进程 P_1和 P_2的源程序如下:

```
%PROS P1()==[
  %STM [
    LB1=START1⇒$OLB1=L11;
    *[LB1=L11⇒($OLB1=L21 | $OLB1=EXIT)
      LB1=L21 ⇒$OC1=0 ∧ $OLB1=L31;
      *[LB1=L31∧(C2=0)⇒($OLB1=L41|$OLB1=EXIT)
        ?[LB1=L41 ∧ (turn=1)⇒($OLB1=L31 | $OLB1=
            L51)
          LB1=l51⇒$OC1=1 ∧ $OLB1=L61
        ]
        *[LB1=L61 ∧ (turn=2)⇒($OLB1=L61'| $OLB1=
            EXIT)
          LB1=L61'⇒$OC1=0∧$OLB1=L31
        ]
      ]
        LB1=L71⇒$OLB1=L81;
      LB1=L81⇒$Oturn=2∧ $OC1=1∧ $OLB1=L91;
      LB1=L91⇒$OLB=L11
    ]
    LB1=L10⇒$OLB1=STOP1
  ]
]
%PROS P2()==[
  %STM [
    LB2=START2⇒$OLB2=L12;
    *[LB2=L12⇒($OLB2=L22 | $OLB2=EXIT)
      LB2=L22⇒$OC2=0 ∧ $OLB2=L32;
      *[LB2=L32 ∧ (C1=0)⇒($OLB2=L42 | $OLB2=EX-
          IT)
      ?[LB2=L42 ∧ (turn=2)⇒($OLB2=L32|$OLB2=L52)
          LB2=l52⇒$OC2=1∧$OLB2=L62
        ]
        *[LB2=L62 ∧ (turn=1)⇒($OLB2=L62'| $OLB2=
            EXIT)
          LB2=L62'⇒$OC2=0 ∧ $OLB2=L32
        ]
      ]
    LB2=L72⇒$OLB2=L82;
    LB2=L82⇒$Oturn=1 ∧ $OC2=1 ∧ $OLB2=L92;
    LB2=L92⇒$OLB=L12
  ]
  LB2=L10⇒$OLB2=STOP2
  ]
]
```

程序的安全性和活性可用如下 XYZ/AE 公式表示。

(1)安全性

$Safety =_{def} \Box\neg(LB_1=L_{71} \wedge LB_2=L_{72})$

(2)活性

$Liveness =_{def} \Diamond((LB_1=L_{71}) \wedge \Diamond(LB_2=L_{72})$

4 语义一致性证明

在本节中,我们将证明前面给出的 XYZ/EE 程序 $P=||[P_1,P_2]$满足性质 Safety 和 Liveness,也即证明如下两个命题:

安全性:$P, LB_1=START_1 \wedge LB_2=START_2, C_1=1 \wedge C_2=1 \wedge turn=1 \models Safety$

活性:$P, LB_1=START_1 \wedge LB_2=START_2, C_1=1 \wedge C_2=1 \wedge turn=1 \models Liveness$

证明:我们首先证明程序 P 满足安全性性质 Safety,然后证明 P 满足活性性质 Liveness。

·安全性

我们可以先证明如下命题 I 成立:

I: $P, LB_1=START_1 \wedge LB_2=START_2, C_1=1 \wedge C_2=1 \wedge turn=1 \models \Box(LB_1=L_{71} \rightarrow (C_1=0))$

显然，由于进程 P_2 不改变变量 C_1 的值，而在进程 P_2 中只有入口为 L_{21} 和 L_{61} 的语句的出口为 L_{31}，而这两条语句的执行都将置 $C_1=0$，因此如下命题 I_1 成立。

I_1：$P, LB_1=START_1 \wedge LB_2=START_2, C_1=1 \wedge C_2=1 \wedge turn=1 \models \Box(LB_1=L_{31} \rightarrow (C_1=0))$

另外，我们可知如下命题 I_2 成立。I_2 可由时序逻辑程序公理 π5 $at\lambda \wedge \neg\lambda \rightarrow \$O at\lambda$ 推论得到，这里可解释为当 $L_{81}=L_{31}$ 时只有进程 P_1 执行入口为 L_{31} 的这条语句才有 $\$O(LB_1=L_{71})$ 成立，而这条语句并不改变 C_1 的值。

I_2：$P, LB_1=START_1 \wedge LB_2=START_2, C_1=1 \wedge C_2=1 \wedge turn=1 \models \Box(((LB_1=L_{31}) \wedge (C_1=0)) \rightarrow \$O(LB_1=L_{71} \rightarrow (C_1=0)))$

由命题 I_1 和 I_2，我们即可知道命题 I 成立。

最后，程序 P 满足安全性的命题可由如下一系列命题推导得出：

(01)$LB_1=START_1 \wedge LB_2=START_2, C_1=1 \wedge C_2=1 \wedge turn=1$

(02)$LB_1=START_1 \wedge LB_2=START_2 \wedge C_1=1 \wedge C_2=1 \wedge turn=1 \rightarrow \Box(LB_1=L_{71} \rightarrow (C1=0))$ //由命题 I 得出

(03)$\Box(LB_1=L_{71} \rightarrow (C_1=0))$ //由(01)、(02)得出

(04)$\Box(LB_1=L_{71} \rightarrow (C_2=1))$ //显然

(05)$\Box(LB_1=L_{71} \rightarrow (C_2=1) \wedge (C_1=0))$ //由(03)、(04)得出

(06)$LB_1=L_{71} \rightarrow (C_2=1) \wedge (C_1=0)$ //由(05)得出

(07)$LB_2=L_{72} \rightarrow (C_1=1) \wedge (C_2=0)$ //由(06)得出

(08)$\Diamond(LB_2=L_{72} \wedge LB_1=L_{71}) \rightarrow \Diamond false$ 由(06)、(07)得出

(09)$\neg\Diamond(LB_2=L_{72} \wedge LB_1=L_{71})$ //由(08)得出

(10)$\Box\neg(LB_2=L_{72} \wedge LB_1=L_{71})$ //显然

就此，我们得到程序 P 满足安全性性质 Safety。

·活性

要证明程序 P 满足活性性质 Liveness，需要我们证明如下两个命题 I_3 和 I_4 成立。由于两个命题是对称的，因此，证明命题 I_3 之后，I_4 也可类似得证。

I_3：$P, LB_1=START_1 \wedge LB_2=START_2, C_1=1 \wedge C_2=1 \wedge turn=1 \models \Diamond(LB_1=L_{71})$

I_4：$P, LB_1=START_1 \wedge LB_2=START_2, C_1=1 \wedge C_2=1 \wedge turn=1 \models \Diamond(LB_2=L_{72})$

在活性的证明过程中，我们需要用到程序的强公平性假设[7]前提。对于 XYZ/E 程序，其强公平性假设是指在程序执行过程中，如果程序某个条件元无线多次使能，则此条件元将被无限多次执行。例如从程序 P 的强公平性假设，我们可得到进程 P_1 满足性质：$\Box\Diamond(LB_1=L_{31} \wedge C_2=1) \rightarrow \Box\Diamond LB_1=L_{71}$，其直观含义为若 P_1 无限次运行到一满足控制标号 $LB_1=L_{31}$ 且 $C_2=1$ 的状态，则 P_1 最终会运行到控制标号 $LB_1=L_{71}$ 的状态。

最后，命题 I_3 的正确性可由如下一系列命题推导得出。这里，整个证明思路采用反证法，其中有些命题可由程序强公平性直接推出，有些命题对应程序不变量的概念，这些不变量比较直观，可用文[8]中关于不变量的证明规则推出。

(01)$\Box(\neg LB_1=L_{71}) \wedge \Box(turn=2) \rightarrow \Diamond LB_1=L_{51}$ //由程序强公平性假设得出

(02)$\Box(LB_1=L_{51} \rightarrow C_1=1)$ //程序不变量

(03)$\Box(\neg LB_1=L_{71}) \wedge \Box(turn=2) \rightarrow \Box(C_1=1)$ //由(01)、(02)得出

(04)$\Box(turn=2) \wedge \Box(C_1=1) \rightarrow \Diamond LB_2=L_{82}$ //由程序强公平性假设得出

(05)$\Box(LB_2=L_{82} \rightarrow turn=1)$ //程序不变量

(06)$\Box(\neg LB_1=L_{71}) \wedge \Box(turn=2) \rightarrow \Diamond(turn=1)$ //由(05)得出

(07)$\Box(\neg LB_1=L_{71}) \rightarrow \Diamond(turn=1)$ 由(06)得出

(08)$\Box(\neg LB_1=L_{71}) \rightarrow \Diamond\Box(turn=1)$ //$\Box\neg(LB_1=L_{71}) \rightarrow \Box(\neg LB_1=L_{81})$，

//而只在执行入口为 $LB_1=L_{81}$ 语

//句后，变量 turn 才会置为2

(09)$\Box\neg(LB_1=L_{71}) \rightarrow \Diamond\Box(LB_1=L_{31} \vee LB_1=L_{41})$ //由程序强公平性假设得出

(10)$\Box(LB_1=L_{31} \vee LB_2=L_{42} \rightarrow (C_1=0) \wedge (C_2=0))$ //程序不变量

(11)$\Box\neg(LB_1=L_{71}) \rightarrow \Diamond\Box(C_1=0 \wedge C_2=0)$ //由(09)、(10)得出

(12)$\Box\neg(LB_1=L_{71}) \rightarrow \Diamond\Box(turn=1 \wedge C_1=0)$ //由(08)、(11)得出

(13)$\Box(turn=1 \wedge C_1=0) \rightarrow \Diamond LB_2=L_{52}$ //由程序强公平性假设得出

(14)$\Box(LB_2=L_{52} \rightarrow C_2=1)$ //程序不变量

(15)$\Box\neg(LB_1=L_{71}) \rightarrow \Diamond\Box(C_2=1)$ //由(12)、(13)、(14)得出

(16)$\Box\neg(LB_1=L_{71}) \rightarrow \Diamond\Box false$ //由(11)、(15)得出

(17)$\neg\Box\neg(LB_1=L_{71})$ //由(16)得出

(18)$\Diamond(LB_1=L_{71})$ //由(17)得出

就此，我们得到程序 P 满足活性性质 Liveness。证毕。

结论 XYZ/E 语言既可以作为规范语言表示系统的静态规范，又可以作为可执行程序描述系统的动态行为，系统对应的可执行程序与其静态规范的语义一致性可在时序逻辑框架下得以证明。本文中我们以 Dekker 算法为例，采用 XYZ/EE 语言实现其算法流程，用 XYZ/AE 语言刻画其安全性和活性，并在时序逻辑框架下证明我们实现的算法与刻画出的性质之间的语义一致性。

参考文献

1 唐稚松，等. 时序逻辑程序设计与软件工程(上册). 科学出版社，1999

2 Tang Chih-Sung. An Outline of XYZ System. In: Logic and Software Engineering (Proc. of Intl. Workshop, Beijing, 1995) (ed. Pnueli A. and Lin H.). Singapore: World Scientific, 1996

3 Stallings W. 操作系统精髓与设计原理(第三版). 清华大学出版社，Prentice-Hall International, Inc, 1998

4 Manna Z, Pnueli A. The Temporal logic of reactive and concurrent systems: Specification. Springer-verlag, New York, 1992

5 Kroger F. Temporal Logic of Programming. Lecture Notes of Tech. Unvi. of Munich, 1982

6 Barringer H R, Gabbay D. Executing Temporal Logic: Review and Prospect. Lecture Notes in Computer Science 335 (ed. By F. H. Vogt), Springer-Verlag, 1988

7 Lamport L. The Temporal Logic of Actions. ACM Trans. Prog. Lang. and Syst., 1994, 16(3): 872～923

8 Manna Z, Pnueli A. Verification of concurrent programs: A temporal proof system. In: J. W. de Bakker, J. Van Leuwen, eds. Foundations of computer Science IV, Distributed Systems: Part 2, pp165-255, Mathematical Centre Tracts 159, Center for Mathematics and Computer Science, Amsterdam, 1983