谓词转换器的相容性*)

陈仪香　孙莉萍

（上海师范大学数学科学学院　上海200234）

摘　要　本文在Dijkstra的卫式语言的基础上，定义了一种特殊的语言，称为弱随机卫式语言，并讨论了该语言所决定的谓词转换器的相容性。

关键词　谓词转换器的相容性，弱随机卫式语言，不确定算子

1　引言

Dijkstra在他的"Discipline of programming"[1]一书中提出了谓词转换器的概念。他把程序看成是一台机器，谓词看成这台机器在工作前后应满足的条件。工作前满足的条件称为前置条件，工作后满足的条件称为后继条件，在纯Hoare逻辑中，这种条件的基本式为{P}S{R}，其中P是前置条件，S是机器，R是后继条件，其表明P是充分条件，它保证了当S执行完毕后，有后继条件R成立。Dijkstra推进了一步，要求R是充要的。因此P可由S和R唯一确定，记为wp(S,R)，它是前置条件称为最弱前置条件。最弱的意思是说，若有另一谓词P′作为前置条件时也能使S的执行终止并满足后继条件R，则定有P′→P。这样wp(S,)可看成是程序S的语义，它是谓词到谓词的函数，故程序S又称为谓词转换器。

本文第一作者在文[2]中讨论了谓词转换器的相容性，引入了相容谓词转换器，建立了它的拓扑语义。本文在Dijkstra的卫式语言的基础上，定义了一种特殊的语言，称为弱随机卫式语言，表明用该语言写成的程序都是相容的，即其谓词转换器是相容的。

2　谓词转换器的相容性

Dijkstra利用最弱前置条件wp(S,)定义了如下三个衡量程序设计语言合理与否的条件，称为健康条件：

H_1：对于任意S有：

wp(S,F)=F.

这就是说任何一种语言的任何一种语句，都不能由某个初始状态出发达到正常终止，而没有计算结果，所以又称做奇迹排除律。

H_2：对于任意S和任意两个后继条件P和Q有：

$P\rightarrow Q$，

则有：

wp(S,P)→wp(S,Q).

H_3：对于任意S和任意两个后继条件P和Q有：

(wp(S,P)∧wp(S,Q))=wp(S,P∧Q).

由健康条件可以推出下面的性质：

H_4：对于任意S和任意两个后继条件P和Q有：

(wp(S,P)∨wp(S,Q))→wp(S,P∨Q).

周巢尘在文[4]中举例表明一般情况下wp(S,P∨Q)$\not\rightarrow$(wp(S,P,)∨wp(S,P)∨wp(S,Q))，其例如下.

$S\equiv if\sim<(x,0)\rightarrow 1\Rightarrow x\ \square\sim<(0,x)\rightarrow 0\Rightarrow x\ fi$ 则

wp(S,x=1)≡(x≥0→T)∧(x≤0→F)≡x>0,

wp(S,x=0)≡(x≥0→F)∧(x≤0→T)≡

*)本文得到国家自然科学基金(69873034)，上海市曙光计划(99SG46)，以及上海市高等学校青年发展基金(98QN)共同资助。
陈仪香　教授，博士，博士后，研究方向：论域理论及其在程序设计语言中的应用。孙莉萍　硕士研究生，专业：应用数学，研究方向：计算机科学中的数学问题。

$x<0$，

而

$$wp(S, x=1 \lor x=0) \equiv T$$

所以

$wp(S, x=1 \lor x=0) \not\longrightarrow wp(S, x=1) \lor wp(S, x=0)$.

若 S 在每时刻可能的结果只有一个，即程序执行图是一线性列，则

H'_4：对于确定 S 和任意两个后继条件 P 和 Q 有：

$$(wp(S,P) \lor wp(S,Q)) = wp(S, P \lor Q).$$

我们关注的语言是不确定性语言，即每个时刻的可能结果不只一个，程序执行图是一树，Dijkatra 引入的卫式语言是这种不确定性语言的核心，我们将在下节中讨论。

定义1 设 P，Q 是两个谓词，若有 $P \land Q = F$，则称 P 与 Q 为不相交的，其并称为不兼容并，记为 $P \sqcup Q$。

定义2 任意机器 S，若对于任意的 P 和 Q，只要 P 与 Q 不相交，就有

$$wp(S, P \sqcup Q) = (wp(S,P) \sqcup wp(S,Q)),$$

则称机器 S 是相容的。

3 弱随机卫式语言

Dijkstra 在文[1]中引入了描述不确定性程序的卫式语言，现叙述在此，这是用 BNF 定义卫式语言语法：

<guarded command>::=<guard>→<guarded list>
<guard>::=<boolean expression>
<guarded list>::=<statement>{;<statement>}
<statement>::=skip|abort|<alternative construct>|<repetitive construct>
<alternative construct>::=if <guarded command set> fi
<repetitive construct>::=do <guarded command set> od
<guarded command set>::=<guarded command>{□<guarded command>}

其中□是卫式命令语言中使用的非确定算子，Dijkstra 的卫式语言的不确定性是由这个算子产生的。如 $B_1 \to SL_1 \square B_2 \to SL_2$，当 B_1 和 B_2 均取值为真时，SL_1，SL_2 都可执行，其可能的结果用 $SL_1(\sigma)$ 和 $SL_2(\sigma)$ 来表示。即使 SL_1，SL_2 仅能产生一个结果，程序 if$B_1 \to SL_1 \square B_2 \to SL_2$fi 也可能拥有两个结果，因此若程序在执行时选择任一结果作为下一步的执行前提，就会产生程序执行的随机性和不确定性。特别当 SL_1，SL_2 产生的结果相互矛盾时，则由于不同的选择会导致完全不同和相反的结果。因此我们应限制这种随机性或不确定性。于是我们要求每个时刻 SL_1 的结果集和 SL_2 的结果集是相交的，即 SL_1 和 SL_2 会产生部分相同的结果，这些部分相同的结果具有 SL_1 和 SL_2 的共性，因此它们作为下一步执行前提的侯选人是恰当的。为此我们引入下面的概念以及一类卫式语言。

定义3 若 $B_1 \to SL_1 \square B_2 \to SL_2 \square \cdots \square B_n \to SL_n$ 中的不确定算子□，对每一时刻 σ，只要 $B_1(\sigma)$ 与 $B_2(\sigma)$ 取真值，就有 $S_1(\sigma) \cap S_2(\sigma) \neq 0$，则称不确定算子□是弱随机的。

对于弱随机的不确定算子□生成的 IF 或 DO 语句，我们可从尽可能多的相交的结果集中选取一个结果是下一步执行的前提。即有尽可能多的 $SL_1, SL_2 \cdots SL_k$，使得 $SL_1 \cap SL_2 \cap \cdots \cap SL_k \neq 0$，此时取 $x \in SL_1 \cap SL_2 \cap \cdots \cap SL_k$ 是下一步执行的前提。

定义4 不确定算子□是弱随机的，此卫式语言就称为弱随机卫式语言。

定理5 用弱随机卫式语言写成的程序是相容的。

下面就此语言的各个命令进行证明。首先做一个说明：当在状态 σ 有谓词 P 成立，则记为 $\sigma \models P$

1）对于"skip"和"abort"命令，相容性可由定义直接得出。

$$wp(skip, P \sqcup Q) = P \sqcup Q = wp(skip,P) \sqcup wp(skip,Q)$$

$$wp(abort, P \sqcup Q) = F = F \sqcup F = wp(abort,P) \sqcup wp(abort,Q)$$

2）对于复合结构，有

$$\begin{aligned} wp(S_1;S_2, P \sqcup Q) &= wp(S_1, wp(S_2, P \sqcup Q)) \\ &= wp(S_1, wp(S_2,P) \sqcup wp(S_2,Q)) = wp(S_1, wp(S_2,P)) \sqcup wp(S_1, wp(S_2,Q)) \\ &= wp(S_1;S_2,P) \sqcup wp(S_1;S_2,Q) \end{aligned}$$

3）对于选择结构

令"IF"是如下的命令

$$if B_1 \to SL_1 \square B_2 \to SL_2 \square \cdots \square B_n \to SL_n fi$$

则

$wp(IF,P \sqcup Q)=wp(IF,P) \sqcup wp(IF,Q)$

证明:σ为任一状态,且

$\sigma \models wp(IF,P \sqcup Q)=BB \wedge (B_1 \rightarrow wp(SL_1,P \sqcup Q)) \wedge \cdots \wedge (B_n \rightarrow wp(SL_n,P \sqcup Q))$

则σ一定满足k个"guard"B_i,即$B_{i_1}(\sigma)=T, B_{i_2}(\sigma)=T, \cdots B_{i_k}(\sigma)=T$故

$$\begin{aligned}\sigma \models\ & BB \wedge wp(SL_{i_1},P \sqcup Q) \wedge \cdots \wedge wp(SL_{i_k},P \sqcup Q)\\ =\ & BB \wedge [wp(SL_{i_1},P) \sqcup wp(SL_{i_1},Q)] \wedge \cdots \wedge wp[(SL_{i_k},P) \sqcup wp(SL_{i_k},Q)]\\ \equiv\ & BB \wedge wp(SL_{i_1},P) \wedge wp(SL_{i_2},P) \wedge \cdots \wedge wp(SL_{i_k},P) \sqcup \cdots\\ & \sqcup (\cdots wp(SL_{i_s},P) \wedge wp(SL_{i_t},Q) \cdots) \sqcup \cdots\\ & \sqcup (BB \wedge wp(SL_{i_1},Q) \wedge wp(SL_{i_2},Q) \wedge \cdots \wedge wp(SL_{i_k},Q))\end{aligned}$$

上式中的第二式是由$SL_{i_1}, SL_{i_2}, \cdots, SL_{i_k}$满足相容性得到的。

下面说明对于任意的s和t,$s \neq t$,$\sigma \not\models BB \wedge wp(SL_{i_s},P) \wedge wp(SL_{i_t},Q)$否则若

$\sigma \models BB \wedge wp(SL_{i_s},P) \wedge wp(SL_{i_t},Q)$

则

$SL_{i_t}\sigma \models Q \quad SL_{i_s}\sigma \models P$

而

$SL_{i_s}\sigma \cap SL_{i_t}\sigma \neq 0$,

所以存在$\sigma' \in SL_{i_s}\sigma \cap SL_{i_t}\sigma$,

$\sigma' \models Q \quad \sigma' \models P$

即

$\sigma' \models P \wedge Q$

这与P,Q不相交矛盾。故有

$\sigma \not\models BB \wedge wp(SL_{i_s},P) \wedge wp(SL_{i_t},Q))$

此时

$$\begin{aligned}\sigma \models\ & (BB \wedge wp(SL_{i_1},P) \wedge wp(SL_{i_2},P) \wedge \cdots \wedge wp(SL_{i_k},P)) \sqcup (BB \wedge wp(SL_{i_1},Q) \wedge wp(SL_{i_2},Q) \wedge \cdots \wedge wp(SL_{i_k},Q))\\ \models\ & wp(IF,P) \sqcup wp(IF,Q)\end{aligned}$$

而另一方面自然成立,故有

$wp(IF,P \sqcup Q)=wp(IF,P) \sqcup wp(IF,Q)$

4)对于循环结构

令"DO"是如下的命令

$do\, B_1 \rightarrow SL_1 \square B_2 \rightarrow SL_2 \square \cdots \square B_n \rightarrow SL_n\, od$

同样也有$wp(DO,P \sqcup Q)=wp(DO,P) \sqcup wp(DO,Q)$

证明:由定义

$wp(DO,P \sqcup Q)=(\exists k, k \geqslant 0, H_k(P \sqcup Q))$

$H_k(P \sqcup Q)=wp(IF,H_{k-1}(P \sqcup Q)) \vee H_0(P \sqcup Q)$

下证

$\forall k \geqslant 0 \quad H_k(P \sqcup Q)=H_k(P) \sqcup H_k(Q)$

当$k=0$时

$$\begin{aligned}H_0(P \sqcup Q)&=(P \sqcup Q) \wedge \sim(\exists j: 1 \leqslant j \leqslant n: B_j)\\ &=[P \wedge \sim(\exists j: 1 \leqslant j \leqslant n: B_j)] \sqcup [Q \wedge \sim(\exists j: 1 \leqslant j \leqslant n: B_j)]\\ &=H_0(P) \sqcup H_0(Q)\end{aligned}$$

假设当$k=n-1$时等式成立,则当$k=n$时

$$\begin{aligned}H_n(P \sqcup Q)&=wp(IF,H_{n-1}(P \sqcup Q)) \vee H_0(P \sqcup Q)\\ &=wp(IF,H_{n-1}(P) \sqcup FH_{n-1}(Q)) \vee H_0(P \sqcup Q)\\ &=wp(IF,H_{n-1}(P)) \sqcup wp(IF,H_{n-1}(Q)) \vee H_0(P) \sqcup H_0(Q))\\ &=H_n(P) \sqcup H_n(Q)\end{aligned}$$

因此,

$$\begin{aligned}wp(DO,P \sqcup Q)&=(\exists k, k \geqslant 0, H_k(P \sqcup Q))\\ &=(\exists k, k \geqslant 0, H_k(P)) \sqcup (\exists k, k \geqslant 0, H_k(Q))\\ &=wp(DO,P) \sqcup wp(DO,Q)\end{aligned}$$

讨论 由上面的讨论可以看出,我们所引入的弱随机卫式语言具有这样的特点:它具有不确定性和随机性,但在具体执行过程中,每一步都可产生一些"好"的结果,这样结果具有一定的共性.选这些结果中的任何一个作为下一步执行的输入状态是令人满意和可接受的,因此这样的程序在随机性方面是相当好的,它介于确定程序和随机程序之间,是可操作和实现的.我们将在另文中研究弱随机语言的论域处理方法。

参考文献

1 Dijkstra E W. A Discipline of programming, Prentice, Prentice-Hall,1976
2 陈仪香.谓词转换器的拓扑语义.数学进展(已接受)
3 屈延文.形式语义学基础与形式说明.科学出版社,1998
4 周巢尘.形式语义学引论.湖南出版社,1985
5 陆汝钤.计算机语言的形式语义.北京:科学出版社,1990