ISSN 1673-9418 CODEN JKYTA8
Journal of Frontiers of Computer Science and Technology
1673-9418/2009/03(02)-0144-10
DOI:10.3778/j.issn.1673-9418.2009.02.003

E-mail:fcst@public2.bta.net.cn
http://www.ceaj.org
Tel:+86-10-51616056

Program: Expressions of Operations on Physical Objects*

YUAN Chongyi[1,2], HUANG Yu[1,2,3+], ZHAO Wen[1,2,3]

1. Key Laboratory of High Confidence Software Technologies of Ministry of Education, Peking University, Beijing 100871, China
2. School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China
3. National Engineering Research Center for Software Engineering, Peking University, Beijing 100871, China

+ Corresponding author: E-mail: hy@pku.edu.cn

程序:物理对象上的操作表达式*

袁崇义[1,2],黄 雨[1,2,3+],赵 文[1,2,3]

1. 教育部高可信软件技术重点实验室,北京 100871
2. 北京大学 信息科学技术学院,北京 100871
3. 北京大学 软件工程国家工程研究中心,北京 100871

摘 要:把赋值语句看作物理对象上的操作时,程序就呈现为物理对象上的操作构成的表达式(简称O表达式)。给出了定义O表达式语法的BNF公式,并用公理规定O表达式的语义。主动式的O表达式以计算最终结果为目的,因而相关公理给出的是施行表达式中操作以后的变量与施行之前变量之间的准确依赖关系。反应式O表达式要对外来需求作反应。描述反应的公理规定如何反应。有关通讯的公理要求正确的信息被正确的接受者收到。共享变量公理则给出有关共享变量的性质判断。例子用于说明异步顺序O表达式的性质是如何分析的。

关键词:程序;物理对象;物理对象上的操作;物理对象上的操作表达式;语义公理

文献标识码:A **中图分类号:**TP311

YUAN Chongyi, HUANG Yu, ZHAO Wen. Program: Expressions of operations on physical objects. Journal of Frontiers of Computer Science and Technology, 2009,3(2):144-153.

* The National Natural Science Foundation of China under Grant No.60803014,60873061(国家自然科学基金); the National High-Tech Research and Development Plan of China under Grant No.2006AA01Z160 (国家高技术研究发展计划(863)); the National Research Foundation for Doctoral Program of Higher Education of China under Grant No.200800011017 (教育部博士点新教师基金).
Received 2008-11, Accepted 2009-01.

Abstract: A program turns out to be an expression of operations on physical objects (operation expression or O_expression for short) when assignments are treated as operations on physical objects. BNF formulas are given to define O_expression syntax while axioms are proposed as semantics of O_expressions. An active O_expression computes some ultimate results. As such, an axiom for active O_expressions defines how a variable after the application of operations in an O_expression is precisely related to variables before the application. A reactive O_expression responses to requests from outside. An axiom, describing responses to requests, tells how a response should be made. Axioms on communications make sure that the right messages to be received by the right receivers while axioms about shared variables depict properties concerning shared variables. Examples are given to show how to analyze properties of asynchronous sequential O_expressions.

Key words: program; physical object; operation on physical object; expression of operations on physical object; semantic axiom

1 Introduction

This paper is the continuation of the one titles <<Assignments: Operations on Physical Object>>, in which a program has turned out to be an expression of operations on physical objects[1]. It devotes itself to a formal treatment of programs based on this observation.

Formal semantics of programs aims at a better understanding and a formal verification of program properties. But, achievements so far in this direction remain a bit far from practical use. The reason is, as far as our understanding, that an assignment in its current form (i.e. $x:=e$ or $x,y:=e_1,e_2$) is not an object that complies with formal treatment. This conclusion is supported by the facts that $wp(x:=e,Q)=Q_e^x$[2] assigns meaning to $x:=e$ in a roundabout way while $x':=e$[3] demands, in order to be semantics of $x:=e$, additional explanation of what x' is. Further investigation has revealed that a program as a whole, consisting of assignments and controls, is as well an object that does not comply with formal treatment. This is why UNITY[4] has removed control mechanisms from its assign section in order to gain full formalism while model checking requires the constructions of a formal model for the program whose properties are to be verified.

The assignment $x:=e$ was viewed in [1] as an operation on physical object x (i.e. the memory location named x) and its takes the form $\bar{x}(e)$ where the over-bar "$\bar{}$" is the operator while x and e are respectively the first and second operands. Now, $\bar{x}(e)$ complies with formal treatment: the semantics of $\bar{x}(e)$ was given in [1] by the axiom $\underline{x}(\bar{x}(e))=e$, or simply $\underline{x}\bar{x}(e)=e$, where the under-bar "_" is another operator on physical object x of which x and $\bar{x}(e)$ are respectively the first and second operands. The second operand of "_" is exactly the program (now an expression of operations on physical objects) whose properties are under investigation. "$\bar{}$" and "_" are respectively the "write-into" and "read-from" operators on x. $\underline{x}(\bar{x}(e))=e$ describes what $\bar{x}(e)$ does while $\forall y:y\neq x\Rightarrow\underline{y}\bar{x}(e)=y$ describes what $\bar{x}(e)$ does not do, or should not do. $\bar{x}(e)$ does not change any variables other than x.

In this paper, we will call an expression of operations on physical object "O_expression". As illustration in [1] with examples, programs in the form of O_expressions do comply with formal treatment. A formal definition of expressions will be proposed in section 2, in terms of BNF (Backus-Naur form).

2 Definition of O_expressions in BNF

Strict BNF formulas contain too many details at the lowest level (e.g. *digit*::=0|1|2|3|4|5|6|7|8|9). To avoid this, it is assumed that the following terminologies have been defined: variable, array, array element, expression (Boolean and mathematical) etc. Further more, we will write $\bar{v}(e)$ to mean $\overline{variable}(expression)$ and $\bar{A}[i](e)$ to mean $\overline{array.element}(expression)$ respectively to simplify formulas to be given. Note also that small letters x,y,i,j,u,v, etc. are variables, capital letters A,B,C,D etc. are array names b,b_1,b_2, are Boolean expression, e,e_1,e_2 are mathematical expressions. Capital E denotes an expression whose value is an array. The index range of arrays is [0..N] ([0..N,0..N] or [0..N][0..N] for 2-dimensional arrays), where N is a positive constant. Small m,n represent positive integer constants.

The arrow symbol "→" will be used to replace conventional "::=" in BNF formulas[4]. Brackets [] indicates one or more appearances of what inside the brackets in BNF. We will leave [] for array indexes and the decorated brackets [‖] will mean one or more appearances.

2.1 BNF Formulas for O_expressions

S-term, i-term, m-term, c-term, ins, soe and aoe are abbreviations for simple term, instance term, multiple term, conditional term, instance sequence, sequential O_expression and asynchronous O_expression respectively. O_expression is abbreviated as oe.

$Special\text{-}term\rightarrow\varepsilon|\varepsilon^{b}$

$s\text{-}term\rightarrow\bar{v}(e)|\bar{A}[i](e)|\bar{A}(E)$

$i\text{-}term\rightarrow\bar{A}[i:b(i)](e(i))|\bar{A}[i,j:b(i,j)](e_1(i,j),e_2(i,j))$

$m\text{-}term\rightarrow s\text{-}term|i\text{-}term|m\text{-}term[|m\text{-}term|]$

$c\text{-}term\rightarrow m\text{-}term^{b}|c\text{-}term|[|c\text{-}term|]$

$term\rightarrow m\text{-}term|c\text{-}term|term[|term|]$

$ins\rightarrow\bar{v}[;i:b(i)](e(i))k|\bar{A}[;i:b(i)](e(i))$

$soe\rightarrow\varepsilon^{b}|term|term|ins|soe[|;soe|]|soe^{b}|soe^{n}|soe^{*}$

$aoe\rightarrow soe[|+soe|]$

$oe\rightarrow soe|aoe|oe[|;oe|]|oe[|+oe|]$

Example 1

$R=\bar{x}(0)\bar{y}(0)\bar{a}(100)\bar{b}(300)$

$P=(\bar{x}(x+1)\bar{a}(a-1)\bar{b}(b-2))^{a\geq 1\wedge b\geq 2}$

$Q=(\bar{y}(y+1)\bar{a}(a-1)\bar{b}(b-4))^{a\geq 1\wedge b\geq 4}$

$S=(\bar{x}(x+2)\bar{y}(y-1)\bar{a}(a-1))^{a\geq 1\wedge b=0}$

$T=(\bar{x}(x-1)\bar{y}(y+1)\bar{b}(b-2))^{a=0\wedge b\geq 2}$

R,P,Q,S,T are m-term, c-term, c-term, c-term and c-term respectively. P^{*},Q^{*},S^{*} and T^{*} are all soe, $P^{*}+Q^{*}$, $S^{*}+T^{*}$ are aoe, and $R;(P^{*}+Q^{*});(S^{*}+T^{*})$ is an oe, i.e. an O_expression.

Example 2

Let p_i,v_i be Boolean variables for $i=1,2,\cdots,N$ and d be a variable of integer.

$R_i=(\varepsilon^{p_i\wedge d>0}\bar{p}_i(0)\bar{d}(d-1))^{*}+(\varepsilon^{v_i}\bar{v}_i(0)\bar{d}(d+1))^{*}$

$R=\bar{d}(1);(R_1+R_2+\cdots+R_N)$

R is designed as a critical section manager to ensure that at most one user is visiting the critical section at any time. R_i takes care of the i-th user who applies for a visit by setting p_i to 1 and informs R_i its leaving by setting v_i to 1.

2.2 Informal Explanation of the Formulas

(1) Special terms ε and ε^{b} play the conventional roles of skip and wait-until-b respectively. ε will not appear in any O_expression as suggested by the formulas. It is used only in semantic axioms. ε^{b} is necessary for synchronization between asynchronous processes.

(2) A simple term assigns a value to a single variable, a single array element or a single array as a whole.

(3) An instance term assigns values to instances of

array elements in a synchronous way. Instances are given by $b(i)$ or $b(i,j)$. The definition of i–term may be extended in an obvious way to include high dimensional arrays, since $D[i][0..N]$ and $D[0..N][i]$ are one–dimensional arrays when i is a constant.

(4) A multiple term assigns values synchronously to one or more variables, including array elements and arrays as a whole.

(5) A conditional term $m\text{–}term^{b}$ functions as m–term when b is true, and as a skip (ε) when $\neg\ b$.

(6) A term assigns values synchronously to all of its constituent variables.

(7) An instance sequence assigns values sequentially to instances of array elements or assigns to a single variable instance of expression $e(i)$, one after another. Instances are ordered by values of the bounded variable i.

3 Axioms on Semantics of O_expression

What follows the arrow symbol "→" in each of the BNF formulas defines a class of O_expressions. Let v be the set of free variables in question.

Definition 1

For O_expression P, the set V_p defined by

$$V_p=\{v|v\in V\wedge \bar{v}\in P\}$$

is called the W–set of P(W stands for write–into), where $\bar{v}\in P$ means that the write–into operation is applied on v in P.

3.1 Axioms on Operators except "+"

Axiom 1 For $v\in V$ and P,Q are O_expressions

(A1.1) $P=Q\Rightarrow \underline{v}(P)=\underline{v}(Q)$.

(A1.2) $v\notin V_p\Rightarrow \underline{v}(P)=v$.

Remarks

(A1.1) asserts that each O_expression changes all variable in V in a unique way. In other words, an O_expression is deterministic.

(A1.2) asserts that P brings no side effect with it.

Axiom 2 P is a simple term

(A2.1) $V_\varepsilon=\phi$.

(A2.2) $\underline{v}\bar{v}(e)=e$.

(A2.3) $\underline{A}[j](\bar{A}[i](e))=e$ if $j=i\sim A[j]$ if $j\neq i$.

(A2.4) $\underline{A}\bar{A}(E)=E$.

Remarks

$V_\varepsilon=\phi$ asserts that ε does not write into any variable. This is different from the convention that $x:=x$ is semantically taken as skip [3].

Axiom 3 Instance term

(A3.1) $\underline{A}[j](\bar{A}[i:b(i)](e(i)))=e(i)$ if $b(j)\sim A[j]$ if $\neg\ b(j)$.

(A3.2) $\underline{A}[l](\bar{A}[i,j:b(i,j)](e_1(i,j),e_2(i,j)))=$

$e_1(l,m)$ if $\exists m:b(l,m)\sim$

$e_2(m,l)$ if $\exists m:b(m,l)\sim$

$A[l]$ if $\forall m:\neg\ (b(l,m)\vee b(m,l))$.

Remarks

(A3.1) may be extended to include high dimensional arrays.

(A1.1) imposes the following constraints on instance terms.

Constraint 1

An i–term $\bar{A}[i,j:b(i,j)](e_1(i,j),e_2(i,j))$ is valid if for any instances l,m_1,m_2 of A, it holds that

$(b(l,m_1)\vee b(l,m_2))\Rightarrow(e_1(l,m_1)=e_1(l,m_2))\wedge$

$(b(m_1,l)\vee b(m_2,l))\Rightarrow(e_2(m_1,l)=e_2(m_2,l))\wedge$

$(b(l,m_1)\vee b(m_2,l))\Rightarrow(e_1(l,m_1)=e_2(m_2,l))$

Axiom 4 P,Q are terms

(A4.1) $\underline{v}(PQ)=\underline{v}(P)$ if $v\in V_P\sim\underline{v}(Q)$ if $v\in V_Q$.

Remarks

A m_term is a special case of terms that does not allow a c_term as its constituent. Thus, (A4.1) covers m_term.

P or Q or both may be a c_term. Semantics of c_terms is given by Axiom 5 below.

In case $v \in V_P \cap V_Q$, $\underline{v}(P)=\underline{v}(Q)$ by (A4.1) and (A1.1).

Axiom 5 c_term

(A5.1) $P^{b}=P$ if $b\sim\varepsilon$ if $\neg\ b$ where P is a term.

Remarks

P if $b\sim\varepsilon$ if $\neg\ b$ is a conditional O_expression that is P when b and ε if not b.

(A5.1) applies as well when P is a soe.

Axiom 6 Instance sequence, $i \in [0..N]$

(A6.1) $\bar{v}[;i:b(i)](e(i))=<;i:b(i)::\bar{v}(e(i))>$.

(A6.2) $\bar{A}[;i:b(i)](e(i))=<;i:b(i)::\bar{A}[i](e(i))>$.

Remarks

On the right of "=" in (A6.1) (and (A6.2)) is a extended quantified expression[4]. Here, ";" is a binary operator in O_expression, but not commutative. Readers not familiar with quantified expressions may think of $<;i:b(i)::\bar{v}(e(i))>$ as an abbreviation of $\bar{v}^{b(0)}(e(0));\bar{v}^{b(1)}(e(1));\cdots;\bar{v}^{b(N)}(e(N))$.

The semantics of operator ";" is given by Axiom 7 below.

Axiom 7 soe, R is a term, P,Q are soe

(A7.1) $\varepsilon^{b}R=R$ if $b\sim\varepsilon^{b}R$ if $\neg\ b$.

(A7.2) $\underline{v}(P;Q)=\underline{v}(Q)(\underline{V}(P)/V)$.

(A7.3) $P^{n}=P^{n-1}$; P when $n>1$.

Remarks

$\varepsilon^{b}R$ implies a deadlock when the Boolean expression b is not true and the soe to which $\varepsilon^{b}R$ belongs is not cooperating with others.

$\underline{v}(Q)(\underline{V}(P)/V)$ represents the expression obtained from $\underline{v}(Q)$ by replacing every $u \in V$ with $\underline{u}(P)$.

It has been proved by symbolic computation that $(P;Q;R)=P;(Q;R)$ is a direct consequence of (A7.2)[1].

(A7.2) applies when P or Q or both P and Q are aoe.

Axiom 8 Operator *

(A8.1) $\underline{v}(P^{*})=\lim_{n\to 0}\underline{v}(P^{n})$.

(A8.2) $P^{*}=P;P^{*}$.

Remarks

When a value is expected from P^{*} in an active O_expression, (A8.1) should be applied. An exception may be raised if the limit does not exist.

(A8.2) applies when a reactive O_expression is in question. No value is expected from P^{*} for any of the variables in this case.

Axiom 8 applies when no asynchronous operator "+" is in relation with P^{*}.

3.2 Asynchronous Operator "+"

The operator "+" connects sequential O_expressions to form an asynchronous O_expression. The ways sequential O_expressions cooperative with each other include synchronous communications via a channel between a sender O_expression and a receiver O_expression, asynchronous communications via a buffer, and variable sharing. A shared variable does not have a fixed writer and/or reader.

3.2.1 Communication Via a Channel

We have proposed axioms in [1] for such communications when it is the only way of message passing. Let us recall it briefly.

Let P, Q be sequential O_expression and $V_P \cap V_Q$ contains no variables other than a communication channel. Let $c \in V_P \cap V_Q$ be the channel between P and Q. The real physical channel may be fixed statically or dynamically, c is the name of it. The sending operation is $\bar{c}(e)$ and the receiving operation is $\bar{u}(c)$ where u is a local variable of the receiver. Let P be the sender and Q be the receiver, then we have

$P=P_1;P_2;\cdots;P_{i-1};\bar{c}(e);P_i;\cdots;P_n$

$Q=Q_1;Q_2;\cdots;Q_{j-1};\bar{u}(c);Q_j;\cdots;Q_m$

$(\bar{c}(e),\bar{u}(c))$ is called a matched pair iff $\bar{c}(e)$ and $\bar{u}(c)$ are respectively the first communication operation in P and Q, i.e $P_1;P_2;\cdots;P_{i-1}$ and $Q_1;Q_2;\cdots;Q_{j-1}$ do not have any terms for sending or receiving.

Axiom 9

(A9.1) $V_P\cap V_Q=\emptyset\Rightarrow\underline{v}(P+Q)=\underline{v}(P)$ if $v\in V_P\sim\underline{v}(Q)$ if $v\in V_Q$.

(A9.2) $V_P\cap V_Q=\{c\}$ and $(\bar{c}(e),\bar{u}(c))$ is a matched pair $\Rightarrow P+Q=P'+Q'$.

Where

$P=P_1;P_2;\cdots;P_{i-1};P_i;\cdots;P_n$

$Q=Q_1;Q_2;\cdots;Q_{j-1};\bar{u}(e');Q_j;\cdots;Q_m$

$e'=e(\underline{v}_P(P_1;P_2;\cdots;P_{i-1})/V_P)$

Remarks

e' is obtained from e by replacing every $x\in V_P$ with $\underline{x}(P_1;P_2;\cdots;P_{i-1})$.

The receiving term was written as $\bar{u}(c)$ in [1] to emphasize the fact that what is really passed via channel c during the communication is the value of e, not the mathematical expression e itself. But $\bar{u}(c)$ is somehow incorrect in our paradigm since a mathematical expression is expected in the place where $\underline{c}$ stands. Thus, $\bar{u}(c)$ is correct for c is a mathematical expression.

It is easy to see from (A9.1) and (A9.2) that

$P+Q=Q+P$

(A9.1) and (A9.2) can be extended to $P+Q+\cdots+R$. Since communication is pairwised. We have proved in [1] that $P+Q+\cdots+R$ is deadlock free as long as every sending and receiving belongs to a matched pair.

Note that when we say synchronous communication is pairwised we do not mean to exclude group communication with one sender and more than one receiver. For example, P is to send e to both Q and R and c_1, c_2 are channels between P and Q and between P and R respectively. We may define $(\bar{c}_1(e)\bar{c}_2(e),\bar{u}(c_1),\bar{v}(c_2))$ to be a matched triple in the way that a matched pair is defined. A matched triple $(\bar{c}_1(e)\bar{c}_2(e),\bar{u}(c_1),\bar{v}(c_2))$ consists of two matched pairs, namely $(\bar{c}_1(e),\bar{u}(c_1))$ and $(\bar{c}_2(e),\bar{v}(c_2))$.

3.2.2 Communications Via a Buffer

Asynchronous communications via a buffer consists of a synchronous communication between the sender and the buffer, a synchronous communication between the buffer and the receiver. The O_expression for the buffer may look like

$(\varepsilon^{b1}\overline{in}(c_1);\cdots)^*+(\varepsilon^{b2}\overline{out}(c_2);\cdots)^*$

Where c_1 is the channel between the sender and the buffer while c_2 is the channel between the buffer and the receiver. Boolean expression b_1 checks whether a sending request is ready while b_2 checks whether a receiving request is ready. Variable *in* is a port for input to the buffer, and *out* is the port for output. Note that the above O_expression does not tell how messages are managed inside the buffer. The two places of $\cdots$ are O_expressions that take care of the management.

Additional axioms are not necessary for communications via buffer.

3.2.3 Variable Sharing

The simplest case of asynchronous O_expression is $P+Q$ where P and Q are terms. P and Q should apply their operations in parallel with each other. Thus, an axiom on $P+Q$ must treat P and Q equally, i.e. $P+Q=Q+P$ should be a direct consequence of the axiom (as suggested by (A7.2)). A candidate axiom on $P+Q$ may be:

Cand_Axiom 1

$P+Q=P$; Q or $P+Q=Q$; P

This axiom is justified since, if P and Q are in read-write conflict, they must be ordered; if P and Q are not in such conflict, $P;Q=Q$; $P=PQ$ (in the sense of (A1.1)). Note PQ is a multiple term and $PQ=QP$.

This axiom covers P^n+Q^m and $P_1;P_2;\cdots;P_n+Q_1;Q_2;\cdots;Q_m$ if it is extended as below:

Cand_Axiom 2

$$P_1;P_2;\cdots;P_n+Q_1;Q_2;\cdots;Q_m=\sigma_1$$
$$\vee P_1;P_2;\cdots;P_n+Q_1;Q_2;\cdots;Q_m=\sigma_2$$
$$\cdots$$
$$\vee P_1;P_2;\cdots;P_n+Q_1;Q_2;\cdots;Q_m=\sigma_l$$

Where $l=\binom{n}{n+m}$ and each of these σ_s is an inter leaving of $P_1;P_2;\cdots;P_n$, $Q_1;Q_2;\cdots;Q_m$ in which $P_1;P_2;\cdots;P_n$ and $Q_1;Q_2;\cdots;Q_m$ keep their orders respectively.

From mathematics we know that $\binom{m}{n+m}=\binom{n}{n+m}$, $P_1;P_2;\cdots;P_n$ and $Q_1;Q_2;\cdots;Q_m$ are equally treated by Cand_Axiom 2.

Can we extend this axiom further to cover P^*+Q^*? For the time being let's assume that P^*+Q^* is an active O_expression. To keep things simple, it is further demanded that P^*+Q^* must comply with the following constraint.

Constraint 2

P^*+Q^* is valid iff:

(1) For every $v\in V$, there exists $\lim\limits_{n\to\infty}\underline{v}(P^n)$ and $\lim\limits_{n\to\infty}\underline{v}(Q^n)$.

(2) The application of operations in P does not destroy the existence of $\lim\limits_{n\to\infty}\underline{v}(Q^n)$ and vis versa.

This constraint does not tell how many times operations in P and Q should be applied to reach these limits. Therefore, there is no way to enumerate all possible interleavings. UNITY defines an execution of an assign section to be an infinite fair interleaving of all assignments in the section [4] and properties shared by all possible executions are properties of the UNITY program. Axiomatic semantics does not care how an O_expression is "executed", it cares only properties deducible from axioms. Thus, we don't follow UNITY approach, do not extend Cand_Axiom 2 to cover P^*+Q^*.

Before going further to propose an axiom for P^*+Q^*, we explain with an example why the second request of constraint 2 is necessary. Let $P=(x+1)^{X<7}$ and $Q=(x-1)^{x\geqslant 7}$. We know form [1] that $\underline{x}(P^*)$ dose exist and $\underline{x}(P^*)\geqslant 7$, $\underline{x}(Q^*)$ exists as well but $\underline{x}(Q^*)<7$. But, if $\underline{x}(P^*+Q^*)$ exists, $\underline{x}(P^*+Q^*)$ must be some integer. One integer can not be both greater than or equal to 7 and smaller than 7 at the same time.

Axiom 10

(A10.1) Cand_Axiom 2.

(A10.2) A property p is a property of P^*+Q^* iff p is a property of P and is also a property of Q.

Remarks

Properties of P^n are also properties of P, the existence of P^* is a property of P as well. The same is true for Q.

Note that it is assumed without saying that P^*+Q^* complies with constraint 2. It is possible that a variable, say x, is only written by P, and a property p refers to only x and no other variables. Is (A10.2) meaningful? Or can we apply (A10.2) to P^*+Q^* and say p is a property of P^*+Q^*? The answer is yes. By (A2.1) and A(1.2), p is also a property of Q. We will

talk about properties of O_expressions in next paper.

To illustrate how (A10.2) is used, let us consider an example. Let be

$P=(\bar{x}(x+1)\bar{a}(a-1)\bar{b}(b-2))^{a\geqslant 1\wedge b\geqslant 2}$

$Q=(\bar{y}(y+1)\bar{a}(a-1)\bar{b}(b-4))^{a\geqslant 1\wedge b\geqslant 4}$

P and Q shares variables a and b.

By (A5.1), (A2.1) and (A1.2) we have

$\underline{x}(P)=x+1$ if $a\geqslant 1\wedge b\geqslant 2\sim x$ if $\neg(a\geqslant 1\wedge b\geqslant 2)$ (1)

$\underline{a}(P)=a-1$ if $a\geqslant 1\wedge b\geqslant 2\sim a$ if $\neg(a\geqslant 1\wedge b\geqslant 2)$ (2)

$\underline{b}(P)=b-2$ if $a\geqslant 1\wedge b\geqslant 2\sim b$ if $\neg(a\geqslant 1\wedge b\geqslant 2)$ (3)

So in any case

$$\underline{x}(P)+\underline{a}(P)=x+a \tag{4}$$

$$2\underline{x}(P)+\underline{b}(P)=2x+b \tag{5}$$

Taking the limit at both sides of (1), (2) and (3), and observe that when $a\geqslant 1\wedge b\geqslant 2$, $\underline{a}(P),\underline{b}(P)$ will decrease, we conclude $\neg(a\geqslant 1\wedge b\geqslant 2)$ will be reached. So $\underline{x}(P^*),\underline{a}(P^*)$ and $\underline{b}(P^*)$ do exist, and

$$\neg(\underline{a}(P^*)\geqslant 1\wedge\underline{b}(Q^*)\geqslant 2) \tag{6}$$

Similarly we can prove that $\underline{y}(Q^*),\underline{a}(Q^*)$ and $\underline{b}(Q^*)$ do exist, and

$$\neg(\underline{a}(Q^*)\geqslant 1\wedge\underline{b}(Q^*)\geqslant 4) \tag{7}$$

It is easy to find that $\underline{a}(P)\leqslant a,\underline{b}(P)\leqslant b$, this guarantees that P does not destroy the existence of $\underline{y}(Q^*)$, $\underline{a}(Q^*)$ and $\underline{b}(Q^*)$. Similarly, Q does not destroy the existence of $\underline{X}(P^*),\underline{a}(P^*)$ and $\underline{b}(P^*)$ neither. So, P^*+Q^* satisfies constraint 2.

Equations (4) and (5) have this counter-part for Q:

$$\underline{y}(Q)+\underline{a}(Q)=y+a \tag{4'}$$

$$4\underline{y}(Q)+\underline{b}(Q)=4y+b \tag{5'}$$

By combine (4) and (4'), (5) and (5') we have:

$\underline{x}(P)+\underline{y}(P)+\underline{a}(P)=x+y+a$

$2\underline{x}(P)+4\underline{y}(P)+\underline{b}(P)=2x+4y+b$

$\underline{x}(Q)+\underline{y}(Q)+\underline{a}(Q)=x+y+a$

$2\underline{x}(Q)+4\underline{y}(Q)+\underline{b}(Q)=2x+4y+b$

By (A10.2) we have:

$$\underline{x}(P^*+Q^*)+\underline{y}(P^*+Q^*)+\underline{a}(P^*+Q^*)=x+y+a \tag{8}$$

$$2\underline{x}(P^*+Q^*)+4\underline{y}(P^*+Q^*)+\underline{b}(P^*+Q^*)=2x+4y+b \tag{9}$$

$R=\bar{x}(0)\bar{y}(0)\bar{a}(100)\bar{b}(300)$ will make $\underline{x}(R)=0$, $\underline{y}(R)=0,\underline{a}(R)=100$ and $\underline{b}(R)=300$. From (A7.2) we conclude that:

$\underline{x}(R;(P^*+R^*))+\underline{y}(R;(P^*+Q^*))+\underline{a}(R;(P^*+Q^*))=100$

$2x(R;(P^*+Q^*))+4y(R;(P^*+Q^*))+b(R;(P^*+Q^*))=300$

and $\underline{a}(R;(P^*+Q^*))\leqslant 0\vee\underline{b}(R;(P^*+Q^*))\leqslant 1$

Since $a=100$ and $b=300$ after R, $\underline{a}(P^n)\geqslant 0$ and $\underline{b}(Q^n)\geqslant 0$, $\underline{a}(Q^n)\geqslant 0$ and $\underline{b}(Q^n)\geqslant 0$ are always true for all n, $n\geqslant 1$. Thus,

$$\underline{a}(R;(P^*+Q^*))=0\vee\underline{b}(R;(P^*+Q^*))=0 \tag{10}$$

Note $\underline{b}(P^n),\underline{b}(Q^n)$ are always even.

There has been a problem for pupils in China: there are 100 heads of chickens and rabbits, and they have 300 legs altogether. How many chickens and rabbits are there? The above O_expression $R;(P^*+Q^*)$ is to count chickens and rabbits: x, y will keep counted numbers of chickens and rabbits respectively and a, b keeps the uncounted heads and legs respectively. R establishes their initial values: $x=0$, $y=0$, $a=100$ and $b=300$. The counting will stop when no head is uncounted or no leg is uncounted (see (10) above).

Example 1 given in section 2 is a complete O_expression for this heads and legs problem. It's easy to prove that (8) and (9) are also properties of S^*+T^* (see example 1).

Thus

$$\underline{x}+\underline{y}+\underline{a}=100 \tag{11}$$

$$2\underline{x}+4\underline{y}+\underline{b}=300 \tag{12}$$

Where $\underline{x}$ stands for $\underline{x}\ (R;(P^*+Q^*);(S^*+T^*))$, so does $\underline{y},\underline{a}$ and $\underline{b}$. Property (10) will be changed to $\underline{a}=0\wedge\underline{b}=0$

by S^*+T^*. So, from (11) and (12), we have:

$\underline{x}+\underline{y}=100$

$2\underline{x}+4\underline{y}=300$

$\underline{x}$ and $\underline{y}$ are solutions to this problem.

Note that if $b<2a \vee b>4a$ initially, the problem would have no solution. S^*+T^* will not end up with $\underline{a}=0 \wedge \underline{b}=0$.

Now, let us pay attention to reactive O_expressions. As a reactive O_expression, P^*+Q^* must take the form

$(\varepsilon^{b1}P)^*+(\varepsilon^{b2}Q)^*$

Where b_1, b_2 are Boolean expressions denoting requests from outside. An example has been given in this section: a buffer. Example 2 in section 2 is also an example of reactive O_expressions: a critical section manager. Although the O_expressions of a reactive system consist of more than one sequential O_expression to react to different outside requests, these sequential O_expression do not interact with each other. Thus, the axiom turns out to be simple.

Axiom 11 On $(\varepsilon^b P)^*$

(A11.1) $b \wedge \neg b(\underline{V}(P)/V) \Rightarrow (\varepsilon^b P)^* = (\varepsilon^b P)^* | (V, \underline{V}(P))$

Where $\neg b(\underline{V}(P)/V)$ is the Boolean expression obtained from $\neg b$ by replacing every variable $v \in V$ with $\underline{v}(P)$. (A11.1) is a refined version of (A8.2).

Remarks

$(V, \underline{V}(P))$ denotes a state in which variable v in V takes $\underline{v}(P)$ as its value. $(\varepsilon^b P)^* | (V, \underline{V}(P))$ indicates that $(\varepsilon^b P)^*$ is carried out at the state $(V, \underline{V}(P))$. The concept of O_expression states will be defined in our next paper on properties of O_expressions.

(A11.1) asserts that when b is true and P changes b from being true to being false $(\varepsilon^b P)^*$ reaches the state $(V, \underline{V}(P))$. The way P changes b indicates that P must have made a response to the outside request represented by b.

(A11.1) is a refined version of (A8.2).

4 General Remarks

Axioms were proposed in [1] for individual operators: "$\bar{}$", "_", ";", "+" and so on. This paper reorganized and refined these axioms with an overall view on O_expresssions, and new axioms (specially 0, 9 and 10) have been proposed, special terms ε and ε^b have also added in the BNF definition of O_expressions.

The write_into operator "$\bar{}$" and the read_from operator "_" were defined in [1] as:

$\bar{}: V \times E_m \rightarrow V_a$

$\_: V \times E_o \rightarrow E_m$

Where E_m stands for "mathematical expression" (including Boolean for Boolean variables), V_a stands for "assigned variables", E_o for O_expression. Now, when we look back at these two definitions from an overall view of O_expressions, we make the following modification:

$\bar{}: V \times E_m \rightarrow E_o$

$\_: V \times E_o \rightarrow E_m$

Axiom(A1.2) has made it possible to remove V_a from them since it is no longer necessary to distinguish an assigned variable from others. Note that $\bar{x}(e)$ was viewed as an assigned variable in [1]. But $\bar{x}(e)$ is in fact an O_expression.

It is important to emphasize again that the concept of physical objects includes not only memory locations, but also all peripherals like line printers, etc. When

wireless communication is concerned, *air* is a physical object as well. Note that *air* is a special class of channels characterized by frequency, and it may have no receiver at all.

Acknowledgement The National Science Foundation has supported our research for many years. The authors are also grateful to the National Research Center for Software Enginee-ring and School of Electronics Engineering and Computer Science of Peking University.

Prof. Yang Fuqing has given the first author personal support ever since the author started working in Peking University. Thanks to Prof. Yang.

Many thanks go to Dr. Gao Xin and Liu Dianxing for their helps.

References:

[1] Yuan Chongyi. Assignment: Operation on a physical object[J]. Journal of Frontiers of Computer Science and Technology, 2008,2(5):487-499.

[2] Hoare C A R. Communicating sequential processes[M]. [S.l.]: Prentice Hall, 1985.

[3] Hoare C A R, He Jifeng. Unifying theories of programming[M]. [S.l.]: Prentice-Hall, 1998.

[4] Chandy K M, Misra J. Parallel program design—a foundation[M]. [S.l.]: Addison-Wesley Publishing Company, 1988.

附中文参考文献:

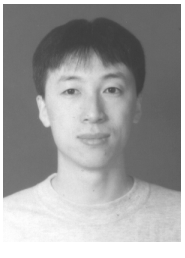
[1] 袁崇义.赋值:物理对象上的操作[J].计算机科学与探索, 2008,2(5):487-499.

YUAN Chongyi was born in 1941. He is a professor and doctoral supervisor at School of Electronics Engineering and Computer Science, Peking University. His research interests include Petri-net theory, parallel program design and program theory.

袁崇义(1941-),男,北京大学信息科学技术学院教授,主要研究领域为 Petri 网,并行程序设计,程序理论。

HUANG Yu was born in 1979. He is an assistant researcher at the National Engineering Research Center for Software Engineering, Peking University. He received the Ph.D. degree in School of Electronics Engineering and Computer Science, Peking University. His re search interests include Petri nets, Web service, software engineering and RFID technology.

黄雨(1979-),男,北京大学软件工程国家工程研究中心助理研究员,2007 年于北京大学信息科学技术学院获理学博士学位,主要研究领域为 Petri 网,Web 服务组合,软件工程,RFID 软件技术。



ZHAO Wen was born in 1967. He is an associate researcher at the National Engineering Research Center for Software Engineering, Peking University. His research interests include software engineering and workflow technology.

赵文 (1967-),男,博士,北京大学软件工程国家工程研究中心副研究员,主要研究领域为软件工程和工作流技术。