EBE: a language for specifying the expected behavior of programs during debugging

NGUYEN HUU CHIEN

1. Introduction

In [1] Bruegge B. and Hibbard P. used *GPE*s (Generalized Path Expression) for specifying expected behavior of programs. *GPE*s are slightly extended version of a *BPE* (Basic Path Expression) with predicates and counters.

A *BPE* is a regular expression with operators sequencing(;), exclusive selection(+) and repetition($*$). The operands, called *PF*s (Path Function), are the names of statements or groups of statements defined in the source program. For each *PF* two counters are defined: the counters *ACT* and *TERM*. These represent the activation and termination number of a *PF* respectively. Predicate is a logical expression involving the counters and the variables of the program and debugger. *BPE* is extended by associating predicates with *PF*s.

In this paper we extended *GPE* by adding the operator shuffle (Δ). This does not increase the power of *GPE*s, but we can describe the expected behavior of a program in a simpler way. In the next sections we define the syntax and semantics of the extended *GPE*s, called *EBE*s (Expected Behavior Expression). The purpose of *EBE*s is to specify the order of execution of *PF*s, the semantics of *EBE*s therefore can be defined by specifying a set of actual behaviors that are valid with respect to a given *EBE*. In section IV we discuss some properties of *EBE*s. According to the syntax and semantics we introduce the syntactical and semantical equivalence of *EBE*s. A sufficient condition for the semantical equivalence of two *EBE*s is given. It is shown that the syntactical equivalence is more powerful than the semantical equivalence. It is also proved that *EBE*s are not more powerful than *GPE*s. In section V we present an implementation of *EBE*s. The implementation is formally defined omitting details of actual implementation, and then its semantics is also defined similarly to that of *EBE*s, that is, by specifying a set of actual behaviors that are valid with respect to a given implementation. Correctness of the implementation is proved by showing a given *EBE* and its implementation recognize the same set of actual behaviors.

In order to make an implementation effective it is necessary to reduce *EBE*s. We give some rules for reducing *EBE*s in section VI.

II. The syntax of EBEs

Assume that the notions ⟨*identifier*⟩, ⟨*integer number*⟩ and ⟨*arithmetic expression*⟩ are known. The other notions are defined in terms of the above ones.

⟨*path function*⟩::=⟨*procedure name*⟩
⟨*procedure name*⟩::=⟨*identifier*⟩
⟨*counter*⟩::=*ACT*(⟨*procedure name*⟩)|*TERM*(⟨*procedure name*⟩)
⟨*counter exp*⟩::=⟨*counter*⟩|⟨*integer variable*⟩|
 ⟨*integer constant*⟩|(⟨*counter exp*⟩)|
 ⟨*counter exp*⟩⟨*binary op*⟩⟨*counter exp*⟩
⟨*binary op*⟩::=+|−|×
⟨*integer variable*⟩::=⟨*identifier*⟩
⟨*integer constant*⟩::=⟨*integer number*⟩
⟨*counter rel*⟩::=⟨*counter exp*⟩⟨*rel*⟩⟨*counter exp*⟩
⟨*arithmetic rel*⟩::=⟨*arithmetic expression*⟩⟨*rel*⟩
 ⟨*arithmetic expression*⟩
⟨*rel*⟩::=$<|>|=|\leqq|\geqq$
⟨*predicate*⟩::=⟨*counter rel*⟩|⟨*arithmetic rel*⟩|(⟨*predicate*⟩)|
 ⟨*predicate*⟩⟨*logic op*⟩⟨*predicate*⟩|¬⟨*predicate*⟩
⟨*logic op*⟩::= ∧ | ∨ | →
⟨*operand*⟩::=⟨*path function*⟩|⟨*path function*⟩[⟨*predicate*⟩]
⟨*EBE*⟩::=⟨*operand*⟩|(⟨*EBE*⟩)|⟨*EBE*⟩;⟨*EBE*⟩|⟨*EBE*⟩+⟨*EBE*⟩|
 ⟨*EBE*⟩*|⟨*EBE*⟩Δ⟨*EBE*⟩

Let E be an *EBE*, we define the language $L(E)$ as follows:
If $E=o$, where o an operand, then $L(E)=\{o\}$. Let $L1=L(E1)$, $L2=L(E2)$, then

$$L(E1;E2) = L1L2,\ L(E1+E2) = L1+L2,\ L(E1*) = L1*,$$

$$L(E1 \Delta E2) = L1 \Delta L2 = \{o_1o'_1 \dots o_no'_n | o_1 \dots o_n \in L1 \text{ and } o'_1 \dots o'_n \in L2, \text{ it may happen that } o_i \text{ and } o'_j \text{ are } \varepsilon\}.$$

Now we give some examples of *EBE*s.

Example.

Initstack; (*Push*[*TERM*(*Push*)−*TERM*(*Pop*)<*N*]+
 Pop[*TERM*(*Push*)−*TERM*(*Pop*)>*o*]+
 Top[*TERM*(*Push*)−*TERM*(*Pop*)>*o*])*.

This *EBE* specifies an expected behavior of the program which states the operational constraints on a bounded stack of length N: first the procedure Initstack has to be called. One of the following can then happen: either procedure *Push* can be called if the size of the stack is smaller than N, or *Top* or *Pop* can be called if the size of the stack is larger than o.

Example. The *EBE*

$$(p;\ q)\Delta(r;\ s)$$

is used to look for activation of the procedure p when p has been called 5 times and the value of the variable A is 4.

Example. The *EBE*

$$p; q\Delta r; s$$

permits possible sequences of the execution of the procedures p, q, r and s as follows:

$$pqrs, prqs, prsq, rpsq, rpqs, rspq.$$

III. The semantics of EBEs

First we define some notions.

Let OB be an arbitrary set (representing a set of all data objects), P a finite set of procedures, and $P' \subset P$.

A *state* is a pair $\langle S, cou \rangle$, where $S \subset OB$, and $cou = \{a_p, t_p | p \in P'\} \subset N+ = \{0, 1, 2, \ldots\}$ (the numbers a_p and t_p represent the activation and termination number of the procedure p), and the "*cou*" is called *counter-state.*

A *concrete (actual) event* is an activation of the procedure p at a state $\langle S, cou \rangle$. We denote it by $e_c = \langle p, S, cou \rangle$.

A *concrete behavior* B is a sequence of concrete events $e_c^1 \ldots e_c^n$. Let $\mathbf{B}$ be the *set* of all concrete behaviors.

A *computational system* is a 5-tuple $\langle OB, P, P', f_a, f_t \rangle$, where f_a and f_t are maps: $\mathbf{B} \to \{g | g$ is function, $g: P' \to N+\}$ which are defined as follows:

The definition of f_a: $f_a(\emptyset)(p) = 0$ for all $p \in P'$,

$$f_a(B\langle p, S, cou \rangle)(p') = f_a(B)(p) + 1 \quad \text{if} \quad p' = p$$
$$= f_a(B)(p') \quad \text{otherwise}, \quad p' \in P', \quad B \in \mathbf{B}.$$

The definition of f_t: $f_t(\emptyset)(p) = 0$ for all $p \in P'$,

$$f_t(B\langle p, S, cou \rangle)(p') = f_t(B)(p) + 1 \quad \text{if} \quad p' = p$$
$$= f_t(B)(p') \quad \text{otherwise}, \quad p' \in P', \quad B \in \mathbf{B} \quad (\emptyset \text{ is the empty sequence}).$$

Let E be an *EBE*, then

$$P_E = \{p | p \text{ is a path function in } E\},$$
$$V_E = \{v | v \text{ is variable in } E, \text{ and } v \neq ACT \text{ and } v \neq TERM\},$$
$$C_E = \{c | c \text{ is constant in } E\}, \quad \text{assume that} \quad C_E \subset OB,$$
$$AT_E = \{ACT(p), TERM(p) | p \in P'\},$$
$$Q_E = \{q | q \text{ is predicate in } E\}.$$

An *abstract event* e_a is a 4-tuple $\langle p, q, V_E, AT_E \rangle$, where $p \in P_E, q \in Q_E$.

An *abstract event expression Ea* of E is an expression obtained as follows. All operands $p[q]$ or p in E are substituted by abstract events $e_a = \langle p, q, V_E, AT_E \rangle$ or $e_a = \langle p, true, V_E, AT_E \rangle$ respectively.

Let $e_c = \langle p, s, cou \rangle$, then the counter-state "*cou*" and the maps f_a and f_t *match* under a concrete behavior B, if $a_p = f_a(Be_c)(p)$, $a_{p'} = f_a(B)(p')$, $p' \in P_E \setminus \{p\}$, and $t_{p'} = f_t(B)(p')$, $p' \in P_E$. This fact is denoted by $Matchs(cou, f_a, f_t, B)$.

An *Interpretation* is a function I: $V_E \cup C_E \cup AT_E \to OB \cup N+$ such that $I(v) \in OB$ for $v \in V_E$, $I(c) \in OB$ for $c \in C_E$, $I(v) \in N+$ for $v \in AT_E$ and I preserves constants and usual arithmetic operators, that is

(1) $I(c) = c$ for all $c \in C_E$,

(2) $I(exp1 \text{ op } exp2) = I(exp1) \text{ op } I(exp2)$, where $op \in \{+, -, \times, /, \uparrow\}$.

A concrete event $e_c = \langle p, S, cou \rangle$ and an abstract event $e_a = \langle p', q, V_E, AT_E \rangle$ match under an interpretation I, if $p = p'$ and $\{I(v) | v \in V_E\} \subset S$ and $I(ACT(p')) = a_{p'}$, $I(TERM(p')) = t_{p'}$ for all $p' \in P_E$. This is denoted by $Matche(e_c, e_a, I)$.

Now we introduce the sets R, BE and EN for Ea. First we supply the abstract events of Ea with indexes 1, 2, ... continuously, in such a manner that any e_a should receive different indexes at different occurrences. If the index of e_a is i, then $e_a(i)$ denotes an *indexed event* of e_a, and the resulting expression is called an *indexed expression* of Ea and denoted $\hat{E}$. Then the sets $R(\hat{E})$, $BE(\hat{E})$ and $EN(\hat{E})$ are defined as follows.

(1) If $\hat{E} = e_a(k)$ then $R(\hat{E}) = \emptyset$, $BE(\hat{E}) = EN(\hat{E}) = \{e_a(k)\}$.

(2) Assume that $Ri = R(\hat{E}i)$, $BEi = BE(\hat{E}i)$ and $ENi = EN(\widehat{Ei})$, $i = 1, 2$, then

$$R(\widehat{E1; E2}) = R1 \cup R2 \cup (EN1 \times BE2); \quad BE(\widehat{E1; E2}) = BE1,$$

$$BE(\widehat{E1*; E2}) = BE1 \cup BE2,$$

$$EN(\widehat{E1; E2}) = EN2, \qquad EN(\widehat{E1; E2*}) = EN1 \cup EN2,$$

$$R(\widehat{E1+E2}) = R1 \cup R2, \qquad BE(\widehat{E1+E2}) = BE1 \cup BE2,$$

$$EN(\widehat{E1+E2}) = EN1 \cup EN2,$$

$$R(\widehat{E1*}) = R1 \cup (EN1 \times BE1), \qquad BE(\widehat{E1*}) = BE1, \quad EN(\widehat{E1*}) = EN1,$$

$$R(\widehat{E1 \Delta E2}) = R1 \cup R2 \cup (\bar{R}1 \times \bar{R}2) \cup (\bar{R}2 \times \bar{R}1)$$

where $\bar{R} = \vec{R} \cup \overleftarrow{R}$, and $\vec{R} = \{a | (a', a) \in R\}$ and $\overleftarrow{R} = \{a | (a, a') \in R\}$,

$$BE(\widehat{E1 \Delta E2}) = BE1 \cup BE2, \qquad EN(\widehat{E1 \Delta E2}) = EN1 \cup EN2.$$

In the following if $(e_a(i), e'_a(k)) \in R(\hat{E})$, then it is written $e_a(i) > e'_a(k)$.

Let $\text{Exp}(\hat{E}) = \{e_a(i) | e_a(i)$ is an indexed event in $\hat{E}\}$.

Let $e_a(i) \in \text{Exp}(\hat{E})$ and $M \subset \text{Exp}(\hat{E})$, then $\vec{e}_a(i) = \{e'_a(k) | e_a(i) > e'_a(k)\}$, and $\vec{M} = \bigcup_{e_a \in M} \vec{e}_a(i)$.

From the construction of the sets $R(\hat{E})$, $EN(\hat{E})$ and $BE(\hat{E})$ it is easy to see the following properties.

Statement 1.

a) $e_a(k) \in BE(\hat{E})$ iff there is a u such that $e_a(k)u \in L(\hat{E})$,
$e_a(k) \in EN(\hat{E})$ iff there is a u such that $ue_a(k) \in L(\hat{E})$,
$e_a(k) > e'_a(n)$ iff there are u, v such that $ue_a(k)e'_a(n)v \in L(\hat{E})$,

b) $e'_a(k_1) > \ldots > e^n_a(k_n)$, $e^1_a(k_1) \in BE(\hat{E})$ iff there is u such that $e^1_a(k_1) \ldots e^n_a(k_n)u \in L(\hat{E})$.

Example. Let $E((p[q]+g[r]);f*)*$. Then

$$Ea = ((e^1_a + e^2_a);\, e^3_a *)*,$$

$$\hat{E} = ((e^1_a(1) + e^2_a(2));\, e^3_a(3)*)*,$$

$$BE(\hat{E}) = \{e^1_a(1), e^2_a(2)\}, \quad EN(\hat{E}) = \{e^1_a(1), e^2_a(2), e^3_a(3)\},$$

$$R(\hat{E}) = \{(e^1_a(1), e^3_a(3)), (e^2_a(2), e^3_a(3)), (e^3_a(3), e^3_a(3)), (e^1_a(1), e^1_a(1)),$$
$$(e^2_a(2), e^2_a(2)), (e^3_a(3), e^1_a(1)), (e^3_a(3), e^2_a(2)), (e^2_a(2), e^1_a(1)), (e^1_a(1), e^2_a(2))\},$$

where $e^1_a = \langle p, q, V_E, AT_E\rangle$, $e^2_a = \langle g, r, V_E, AT_E\rangle$ $e^3_a = \langle f, \text{true}, V_E, AT_E\rangle$.

Definition. Let $R = \langle OB, P, P', f_a, f_t\rangle$ be a computational system and E an *EBE* such that $P' = P_E$. The *semantics* of E is defined by the predicate $Valid_E$: $\mathbf{B} \to \to \{\text{true, false}\}$ with the partial map $Next_E$: $\mathbf{B} \to \{\vec{M} \mid M \subset Exp(\hat{E})\}$, in such a way that $Next_E(B)$ is defined iff $Valid_E(B)=$true. The $Valid_E$ and $Next_E$ are defined recursively as follows.

(1) Let $e_c = \langle p, S, cou\rangle$, then $Valid_E(e_c) = Matchs(cou, f_a, f_t, \emptyset) \& M \neq \emptyset$, where

$$M = \{e_a(i) | e_a(i) \in BE(\hat{E}) \,\&\, e_a = \langle p, q, V_E AT_E\rangle \,\&\, (\exists I)(Matche(e_c, e_a, I) \,\&\, Sat(q, I))$$
$$= true\}$$

(*Sat* is defined later). And $Next_E(e_c)$ *is* defined iff $Valid_E(e_c)=$true, and then $Next_E(e_c) = \vec{M}$.

(2) Let $e_C = \langle p, S, cou\rangle$ and $B \in \mathbf{B}$, then

$$Valid_E(Be_c) = Valid_E(B) \& Next_E(B) = \vec{N} \& Matchs(cou, f_a, f_t, B) \& M \neq \emptyset,$$ where

$$M = \{e_a(i) | e_a(i) \in \vec{N} \& e_a = \langle p, q, V_E, AT_E\rangle \& (\exists I)(Matche(e_c, e_a, I) \& Sat(q, I)) = \text{true}\}.$$

And $Next_E(Be_c)$ is defined iff $Valid_E(Be_c)=$true, and then $Next_E(Be_c) = \vec{M}$.

The definition of the predicate Sat. $Sat(q, I)$ is defined according to the syntax of the predicate q.

$$Sat(\langle counter\ exp\rangle \langle rel\rangle \langle counter\ exp\rangle, I) =$$
$$= I(\langle counter\ exp\rangle) \langle rel\rangle I(\langle counter\ exp\rangle)$$

$$Sat(\langle arithmetic\ exp\rangle \langle rel\rangle \langle arithmetic\ exp\rangle, I) =$$
$$= I(\langle arithmetic\ exp\rangle) \langle rel\rangle I(\langle arithmetic\ exp\rangle)$$

$$Sat(\langle predicate\rangle \langle logic\ op\rangle \langle predicate\rangle, I) =$$
$$= Sat(\langle predicate\rangle, I) \langle logic\ op\rangle Sat(\langle predicate\rangle, I)$$

$$Sat(\neg \langle predicate\rangle, I) = \neg Sat(\langle predicate\rangle, I).$$

Let $\mathbf{B}(E)=\{B|B\in\mathbf{B}$ and $Valid_E(B)=\text{true}\}$.

From the definition of the semantics of *EBE*s it is easy to see the following fact.

Fact 1. Let $Bn=e_c^1\dots e_c^n$, $e_c^i=\langle p_i, S_i, cou_i\rangle$, $i=1, \dots, n$, then

$Valid_E(Bn) = \text{true}$ iff

$Matchs(cou_i, f_a, f_t, B_{i-1})$, $i = 1, \dots, n$, $B_0 = \emptyset$, and there is a sequence $\{Mi\}_{i=1}^n$ such that

$$Mi = \{e_a(k)|e_a(k)\in\vec{M}_{i-1}\,\&\, e_a =$$
$$= \langle p_i, q, V_E, AT_E\rangle \,\&\, \exists I(Matche(e_c^i, e_a, I)\,\&\, Sat(q, I) = \text{true}\} \neq \emptyset,$$

and $Next_E(Bi) = \overrightarrow{Mi}$, $i = 1, \dots, n$, $\overrightarrow{Mo} = BE(\hat{E})$.

IV. Some properties of EBE

Definition. Two *EBE*s E and E' are *syntactically equivalent* iff $L(E)=L(E')$.

Definition. Two *EBE*s E and E' are *semantically equivalent* iff $\mathbf{B}(E)=\mathbf{B}(E')$.

Theorem 1. If E and E' are *EBE*s such that $L(E)\subset L(E')$ and for all $u\in L(E')\setminus L(E)$ there are $v\in L(E)$ and w for which $v=uw$ then E and E' semantically equivalent.

Proof. According to the construction of *Ea* we can identify *Ea* with *E*, thus *L*(*Ea*) with *L*(*E*). First we prove the following facts.
For any E and $Bn=e_c^1\dots e_c^n$, $e_c^i=\langle p_i, S_i, cou_i\rangle$.

Fact 2. If there is a sequence $\{Mi\}_{i=1}^n$ such that

$$Mi = \{e_a(k)|e_a(k)\in\vec{M}_{i-1}\,\&\, e_a =$$
$$= \langle p_i, q, V_E, AT_E\rangle \,\&\, \exists I(Matche(e_c^i, e_a, I)\,\&\, Sat(q, I)) = \text{true}\} \neq \emptyset,$$
$$i = 1, \dots, n,\ \overrightarrow{Mo} = E(B\hat{E}),$$

then there is a sequence $\{e_a^i(k_i)\}_{i=1}^n$, $e_a^i=\langle p_i, q_i, V_E, AT_E\rangle$, for which $e_a^i(k_i)\in Mi$, $i=1, \dots, n$ and $e_a^1(k_1)>\dots>e_a^n(k_n)$.

The existence of the desired sequence is shown by induction as follows.
Since $Mn\neq\emptyset$, thus there is an $e_a^n(k_n)\in Mn$, $e_a^n=\langle p_n, q_n, V_E, AT_E\rangle$. From the definition of *Mn* there is an $e_a^{n-1}(k_{n-1})\in M_{n-1}$ for which $e_a^{n-1}(k_{n-1})>e_a^n(k_n)$, $e_a^{n-1}= =\langle p_{n-1}, q_{n-1}, V_E, AT_E\rangle$. Assume that the sequence $\{e_a^j(k_j)\}_{j=i}^n$, $i>1$, is constructed. Then from the definition of *Mi* there is an $e_a^{i-1}(k_{i-1})\in M_{i-1}$ for which $e_a^{i-1}(k_{i-1})>e_a^i(k_i)$. So we get the desired sequence.

Fact 3. If there is a sequence $\{e_a^i(k_i)\}_{i=1}^n$, $e_a^i=\langle p_i, q_i, V_E, AT_E\rangle$, such that there is a u for which $e_a^1(k_1)\dots e_a^n(k_n)u\in L(\hat{E})$, and for each $i\leqq n$ there is an I for which *Matche*(e_c^i, e_a^i, I) *and* *Sat*(q_i, I)=true, then $e_a^i(k_i)\in Mi$, $i=1, \dots, n$ (Mi is defined in Fact 2, $i=1, \dots, n$).

This can easily be proved by induction on $i\leqq n$ (using Statement 1).

Now we prove Theorem 1.

We have to prove that $\mathbf{B}(E)=\mathbf{B}(E')$.

From Fact 1 it is sufficient to prove that for any $Bn=e_c^1\dots e_c^n$, $e_c^i=\langle p_i, S_i, cou_i\rangle$, $i=1, \dots, n$, the following holds.

$$(+)\begin{cases} Matchs(cou_i, f_a, f_t, B_{i-1}),\ i=1,\dots,n,\ Bo=\emptyset, \text{ and there is} \\ \text{a sequence } \{Mi\}_{i=1}^n \text{ such that} \\ Mi=\{e_a(k)|e_a(k)\in\vec{M}_{i-1}\,\&\,e_a=\langle p_i, q, V_E, AT_E\rangle\,\&\,\exists I(Matche(e_c^i, e_a, I)\,\& \\ \qquad Sat(q, I))=\text{true}\}\neq\emptyset, \\ \text{and } Next_E(Bi)=\vec{Mi},\ i=1,\dots,n,\ \vec{Mo}=BE(\hat{E}). \end{cases}$$

iff

$$(++)\begin{cases} Matchs(cou_i, f_a, f_t, B_{i-1}),\ i=1,\dots,n,\ Bo=\emptyset, \text{ and there is} \\ \text{a sequence } \{Ni\}_{i=1}^n \text{ such that} \\ Ni=\{e_a(1)|e_a(1)\in\vec{N}_{i-1}\,\&\,e_a=\langle p_i, q, V_{E'}, AT_{E'}\rangle\,\&\,\exists I(Matche(e_c^i, e_a, I)\,\& \\ \qquad Sat(q, I))=\text{true}\}\neq\emptyset, \\ \text{and } Next_{E'}(Bi)=\vec{Ni},\ i=1,\dots,n,\ \vec{No}=BE(\hat{E}'). \end{cases}$$

This is shown by induction on n.

1) It is easy to show that the statement holds for $n=1$.

2) Assume that the statement holds for n. Now we prove that the statement holds for $n+1$ too.

(+)⇒(++). Assume that (+) holds for $n+1$. Then (++) holds for n. We have yet to prove that $N_{n+1}\neq\emptyset$ and $Next_{E'}(B_{n+1})=\vec{N}_{n+1}$.

According to Fact 2 there is a sequence $\{e_a^i(k_i)\}_{i=1}^{n+1}$, $e_a^i=\langle p_i, q_i, V_E, AT_E\rangle$, for which $e_a^i(k_i)\in Mi$, $i=1, \dots, n+1$, and $e_a^1(k_1)>\dots>e_a^{n+1}(k_{n+1})$. Since $e_a^1(k_1)\in BE(\hat{E})$, thus, by Statement 1, there is a u for which $e_a^1(k_1)\dots e_a^{n-1}(k_{n+1})u\in L(\hat{E})$ which implies that there is a v for which $e_a^1\dots e_a^{n+1}v\in L(Ea)$. Since $L(Ea)\subset L(E'a)$, thus $e_a^1\dots e_a^{n+1}v\in L(E'a)$ which implies that there are a sequence $\{l_i\}_{i=1}^{n+1}$ and a u' for which $e_a^1(l_1)\dots e_a^{n+1}(l_{n+1})u'\in L(\hat{E}')$. Then, by Fact 3, we have $e_a^i(l_i)\in Ni$, $i=1, \dots, n+1$. So $N_{n+1}\neq\emptyset$. Since (++) holds for n, thus $Next_{E'}(Bn)=N_n$ and, by Fact 1, $Valid_{E'}(Bn)$=true, therefore $Valid_{E'}(B_{n+1})$=true (by the definition of Semantics of *EBE*s) which implies $Next_{E'}(B_{n+1})$ is defined and is $\vec{N}_{n+1}$.

(++)⇒(+). Assume that (++) holds for $n+1$. Then (+) holds for n. We have yet to prove that $M_{n+1}\neq\emptyset$ and $Next_E(B_{n+1})=\vec{M}_{n+1}$. Similarly to the above argument we have the sequence $\{e_a^i(k_i)\}_{i=1}^{n+1}$, $e_a^i=\langle p_i, q_i, V_{E'}, AT_{E'}\rangle$, for which $e_a^i(k_i)\in Ni$, $i=1, \dots, n+1$, and there is a v such that $e_a^1\dots e_a^{n+1}v\in L(E'a)$. We have two cases:

$$\text{either}\quad e_a^1\dots e_a^{n+1}v\in L(Ea)$$

$$\text{or}\quad e_a^1\dots e_a^{n+1}v\in L(E'a)\setminus L(Ea).$$

In the second case there is a v' for which $e_a^1 \dots e_a^{n+1} vv' \in L(Ea)$. So in both cases we have that there are a sequence $\{l_i\}_{i=1}^{n+1}$ and w for which $e_a^1(l_1) \dots e_a^{n+1}(l_{n+1}) w \in L(\hat{E})$. Therefore by Fact 3 $e_a^i(l_i) \in Mi$, $i=1, \dots, n+1$. So $M_{n+1} \neq \emptyset$. Again by the same argument seen above we get that $Next_E(B_{n+1}) = \vec{M}_{n+1}$.

Theorem 2. a) if E and E' are syntactically equivalent then they are semantically equivalent too.

b) There exist two *EBE*s E and E' which are semantically equivalent but not syntactically equivalent.

Proof. a) It is a corollary of Theorem 1.

b) In order to prove this we give an example.

Let $E = p_1 + (p_1; p_2)$ and $E' = p_1; p_2$, it is clear that E and E' satisfy Theorem 1, therefore E and E' are semantically equivalent but not syntactically equivalent because $L(E) \neq L(E')$.

An *EBE* is a GPE *(Generalised Path Expression)* if the operator Δ does not occur in it.

Theorem 3. For every *EBE* E there exists a *GPE* E' such that E and E' are semantically equivalent.

Proof. First we construct an automaton M for which $L(M) = L(E)$. In order to do this we define the sets $R(\hat{E})$, $BE(\hat{E})$ and $EN(\hat{E})$ similarly to those of Section III. The automaton $M = (\Sigma, St, s_0, \delta, F)$ is then constructed as follows. Let $\Sigma = \{e_a | e_a$ is in $Ea\} = \{e_a^1, \dots, e_a^n\}$. Let s_0 be an arbitrary symbol. Then $\delta(s_0, e_a^i) = \{e_a^i(k) | e_a^i(k) \in BE(\hat{E})\} = s_1^i$. So we have defined states $s_0, s_1^1, s_1^2, \dots, s_1^n$ of St. Suppose that a state s of St is defined, then

$$\delta(s, e_a^i) = \{e_a^i(k) | \exists e_a^j(m) (e_a^j(m) \in s \,\&\, e_a^j(m) > e_a^i(k)) = \text{true}\}, \quad i = 1, 2, \dots, n.$$

Finally let

$$F' = \{s | s \cap EN(\hat{E}) \neq \emptyset\} \quad \text{and} \quad F = F' \cup \{s_0\} \quad \text{if} \quad \varepsilon \in L(Ea) = F,$$

and $F = F'$, otherwise.

It is easy to see that $L(M) = L(Ea)$. It is known that there is a regular expression E' over Σ for which $L(M) = L(E')$. Thus we have $L(Ea) = L(E')$. From the construction of Ea we can identify Ea with E, thus $L(Ea)$ with $L(E)$. Therefore, by Theorem 2, E and E' are semantically equivalent.

V. Implementation of EBE

The implementation of *EBE* is defined using the concept of automaton.

Let $R = \langle OB, P, P', f_a, f_t, \rangle$ be a computational system. Let $Q = \{q | q$ is predicate, $q: OB' \times \{f_a(B)(p) | B \in \mathbf{B}, p \in P'\}^m \times \{f_t(B)(p) | B \in \mathbf{B}, p \in P'\}^n \to \{\text{true, false}\}$,

and $M = (\Sigma, St, s_0, \delta, F)$ a deterministic finite automaton, where $\Sigma \subset P' \times Q$. For all $p \in P'$ the set Condition $(p) = \{\langle s, q \rangle | \delta(s, \langle p, q \rangle)$ is defined$\}$ is called *condition* of the procedure p.

Definition. An *Implementation* is a set $I(M)=\{\langle p, Condition(p)\rangle | p\in P'\}$. For simplicity we often omit the argument M.

Restriction.

It is assumed about the automaton M that if $s_i=\delta(s_{i-1}, b), b_i=\langle p_i, q_i\rangle$, $b_i=\langle p_i, q_i\rangle$, $i=1, 2, ..., n$, then there is a u such that $b_1 b_2 ... b_n u\in L(M)$.

Now we define the semantics of Implementations.

Definition. Let I be an Implementation. The *semantics* of I is defined by the predicate $Valid_I$ with the partial map $Next_I$, where $Valid_I$: $\mathbf{B}\to\{\text{true, false}\}$ and $Next_I$: $\mathbf{B}\to 2^{St}$, in such a way that $Next_I(B)$ is defined iff $Valid_I(B)=\text{true}$. The $Valid_I$ and $Next_I$ are defined recursively as follows.

(1) Let $e_c=\langle p, S, cou\rangle$ be an actual event, then

$$Valid_I(e_c) = Matchs(cou, f_a, f_t, \emptyset) \,\&\, \exists\langle s_0, q\rangle(\langle s_0, q\rangle\in Condition(p) \,\&\, Sat(q, e_c, \emptyset)),$$

(Sat is defined later).

$Next_I(e_c)$ is defined iff $Valid_I(e_c)=\text{true}$, and then
$Next_I(e_c) = \{s | s\in St \,\&\, \exists q(q\in Q \,\&\, s=\delta(s_0,\langle p, q\rangle) \& Sat(q, e_c, \emptyset))=\text{true}\}$.

(2) Let $e_c=\langle p, S, cou\rangle$ and $B\in\mathbf{B}$, then

$$Valid_I(Be_c) = Valid_I(B) \,\&\, Next_I(B) =$$

$$= G \,\&\, Matchs(cou, f_a, f_t, B) \,\&\, \exists s\exists q(s\in G \,\&\, q\in Q \,\&\, \langle s, q\rangle\in Condition(p) \,\&$$

$$Sat((q, e_c, B)).$$

$Next_I(Be_c)$ is defined iff $Valid_I(Be_c)=\text{true}$, and then $Next_I(Be_c)=H$, where $H=\{s | s\in St \,\&\, \exists s'\exists q(s'\in G \,\&\, q\in Q \,\&\, s=\delta(s', \langle p, q\rangle) \,\&\, Sat(q, e, B))=\text{true}\}$.

The definition of Sat. $Sat(q, e_c, B)$ is defined simply as follows.

$$\text{Sat}(q, e_c, B) = q(S, f_a(Be_c)(p), \{f_a(B)(p') | p'\in P'\setminus\{p\}\}, \{f_t(B)(p') | p'\in P'\}).$$

Similarly to Fact 1 it is easy to see the following fact (from the definition of the semantics of Implementation).

Fact 4. For any $Bn=e_c^1 ... e_c^n$, $e_c^i=\langle p_i, S_i, cou_i\rangle$, $Valid_I(Bn)=\text{true}$ iff

$Matchs(cou_i, f_a, f_t, B_{i-1})$, $i=1, ..., n$, $Bo=\emptyset$, and there is a sequence $\{H_i\}_{i=1}^n$ so that

$$Hi = \{s | \exists s'\exists q(s'\in H_{i-1} \,\&\, q\in Q \,\&\, s = \delta(s', \langle p_i, q\rangle) \,\&\, \text{Sat}(q, e_c^i, B_{i-1})) = \text{true}\} \neq \emptyset,$$

and $Next_I(Bi)=Hi$, $i=1, ..., n$, $Ho=\{s_0\}$.

Let $\mathbf{B}(I)=\{B | B\in\mathbf{B}$ and $Valid_I(B)=\text{true}\}$.

Definition. An *Implementation of an EBE E* is an Implementation

$$I = \{\langle p, Condition(p)\rangle | p\in P'\} \text{ such that } P' = P_E \text{ and } \mathbf{B}(I) = \mathbf{B}(E).$$

Now we give an algorithm for transforming an *EBE E* to its Implementation.

Algorithm.

1. Transforming E to the following: we substitute all operands $p[q]$ or p of E by $e=\langle p, q\rangle$ or $e=\langle p, \text{true}\rangle$ respectively. The resulting expression is denoted by Ee.

2. From Ee constructing an automaton $M=\langle \Sigma, St, s_0, \delta, F\rangle$ as that of Theorem 3, where $\Sigma=\{e|e \text{ is in } Ee\}$.

3. For all $p\in P_E$ constructing the set $Condition(p)$, obtaining

$$I = \{\langle P, Condition(p)\rangle | p\in P_E\}.$$

Theorem 4. I is an Implementation of E.

Proof. First we prove the following facts. For any implementation I and actual behavior $Bn=e_c^1 \dots e_c^n$, $e_c^i=\langle p_i, S_i, cou_i\rangle$

Fact 5. If there is a sequence $\{Hi\}_{i=1}^n$ such that

$$Hi = \{s|\exists s' \exists q(s'\in H_{i-1} \& q\in Q \& s = \delta(s', \langle p_i, q\rangle) \& \mathrm{Sat}(q, e_c^i, B_{i-1})) = \text{true}\} \neq \emptyset,$$

$$i = 1, \dots, n, \quad Ho = \{s_0\},$$

then there are sequences $\{s_i\}_{i=0}^n$ and $\{e_i\}_{i=1}^n$, $e^i=\langle p_i, q_i\rangle$, for which $s_i\in Hi$, $s_i=\delta(s_{i-1}, e^i)$ and $Sat(q_i, e_c^i, B_{i-1})=\text{true}$, $i=1, \dots, n$.

This can be proved by induction as follows. Since $Hn\neq\emptyset$, there is an $s_n\in Hn$. From the definition of Hn there are $s_{n-1}\in H_{n-1}$ and $e^n=\langle p_n, q_n\rangle$ for which $s_n=\delta(s_{n-1}, e^n)$ and $Sat(q_n, e^n, B_{n-1})=\text{true}$. Assume that the sequences $\{s_j\}_{j=i}^n$ and $\{e^j\}_{j=i+1}^n$, are constructed. Then from the definition of Hi there are $s_{i-1}\in H_{i-1}$ and $e^i=\langle p_i, q_i\rangle$, for which $s_i=\delta(s_{i-1}, e^i)$, and $Sat(q_i, e_c^i, B_{i-1})=\text{true}$. So we get the desired sequences.

Fact 6. If there are sequences $\{s_i\}_{i=0}^n$ and $\{e^i\}_{i=1}^n$, $e^i=\langle p_i, q_i\rangle$, for which $s_i=\delta(s_{i-1}, e^i)$ and $Sat(q_i, e^i, B_{i-1})=\text{true}$, $i=1, \dots, n$, then $s_i\in Hi$, $i=0, 1, \dots, n$ (Hi is defined in Fact 5).

This can easily be proved by induction on $i\leq n$.

Now we prove the Theorem. It is easy to see that:

1. The automaton M satisfies the Restriction.
2. $L(M)=L(Ee)$.

Now we show that $\mathbf{B}(E)=\mathbf{B}(I)$. By Fact 1 and Fact 4 it is sufficient to prove that for any EBE E and Implementation I if $Bn=e_c^1 \dots e_c^n$, $e_c^i=\langle p_i, S_i, cou_i\rangle$, $i=1, 2, \dots, n$, then

$$(*)\begin{cases} Matchs(cou_i, f_a, f_t, B_{i-1}),\ i = 1, \dots, n,\ Bo = \emptyset, \text{ and there is a} \\ \text{sequence } \{Mi\}_{i=1}^n \text{ such that} \\ Mi = \{e_a(k)|e_a(k)\in \vec{M}_{i-1} \& e_a^i = \langle p_i, q, V_E, AT_E\rangle \& \exists \text{ interpretation } I \\ (Matche(e_c^i, e_a, I) \& Sat(q, I)) = \text{true}\} \neq \emptyset, \text{ and } Next_E(Bi) = \overrightarrow{Mi}, \\ i = 1, \dots, n, \quad \overrightarrow{Mo} = BE(\hat{E}) \end{cases}$$

iff

$$(**)\begin{cases} Matchs(cou_i, f_a, f_t, B_{i-1}),\ i = 1, \dots, n, \text{ and there is a} \\ \text{sequence } \{Hi\}_{i=1}^{n} \text{ such that} \\ Hi = \{s | s \in St \,\&\, \exists s' \exists q (s' \in H_{i-1} \,\&\, q \in Q \,\&\, s = \delta(s', \langle p_i, q \rangle) \,\&\, Sat(q, e_c^i, B_{i-1})) \\ \quad = \text{true}\} \neq \emptyset, \text{ and } Next_I(Bi) = Hi, \quad i = 1, \dots, n, \quad Ho = \{s_0\}. \end{cases}$$

From the construction of *Ea* and *Ee* we can identify *Ea* with *Ee* and therefore $L(Ea)$ with $L(Ee)$ too, so $L(Ea) = L(Ee) = L(M)$.

Now we prove that (*) iff (* *) for any *Bn*. This is shown by induction on *n*.

(1) It is easy to see that the statement holds for $n=1$.

(2) Assume that the statement holds for *n*.

$(*)\Rightarrow(**)$. Suppose that (*) holds for $n+1$. Then (* *) holds for *n*. We have yet to prove that $H_{n+1} \neq \emptyset$, and $Next_I(B_{n+1}) = H_{n+1}$.

By Fact 2 we have a sequence $\{e_a^i(k_i)\}_{i=1}^{n+1}$, $e_a^i = \langle p, q_i, V_E, AT_E \rangle$, such that $e_a^1(k_1) > \dots > e_a^{n+1}(k_{n+1})$, and $e_a^i(k_i) \in Mi$, $i=1, \dots, n+1$. Therefore according to Statement 1 there is a *u* for which $e_a^1(k_1) \dots e_a^{n+1}(k_{n+1}) u \in L(\hat{E})$ which implies that there is a *v* such that $e_a^1 \dots e_a^{n+1} v \in L(Ea)$. Since $L(Ea) = L(M)$ thus there exists a sequence $\{s_i\}_{i=0}^{n+1}$ such that $s_i = \delta(s_{i-1}, e^i)$, where $e^i = \langle p_i, q_i \rangle$, $i=1, \dots. n+1$. It is easy to see that for all $i \leq n+1$, $Sat(q_i, e_c^i, B_{i-1}) = \text{true}$ (from the definition of *Matche, Matchs,* interpretation *I*, $Sat(q, I)$ and $Sat(q, e_c, B)$). So by Fact 6 we have $s_i \in Hi$, $i=1, \dots, n+1$, that is $H_{n+1} \neq \emptyset$. Since (* *) holds for *n* thus $Next_I(Bn) = Hn$ and, by Fact 4, $Valid_I(Bn) = \text{true}$. Therefore from the definition of the Semantics of Implementation $Valid_I(B_{n+1}) = \text{true}$ which implies that $Next_I(B_{n+1})$ is defined and is H_{n+1}.

$(**)\Rightarrow(*)$. Assume that (* *) holds for $n+1$. Then (*) holds for *n*. We have yet to prove that $M_{n+1} \neq \emptyset$ and $Next_E(B_{n+1}) = \vec{M}_{n+1}$.

According to Fact 5 we have the sequence $\{s_i\}_{i=0}^{n+1}$ and $\{e^i\}_{i=1}^{n+1}$ for which $s_i \in Hi$, $s_i = \delta(s_{i-1}, e^i)$, $Sat(q_i, e_c^i, B_{i-1}) = \text{true}$, and $e^i = \langle p_i, q_i \rangle$, $i=1, \dots, n+1$. Then, by Restriction, there is a *u* for which $e^1 \dots e^{n+1} u \in L(M) = L(Ea)$ which implies that there are a *v* and a sequence $\{k_i\}_{i=1}^{n+1}$ for which $e_a^1(k_1) \dots e_a^{n+1}(k_{n+1}) v \in L(\hat{E})$, $e_a^i = \langle p_i, q_i, V_E, AT_E \rangle$. It is easy to see that for each $i \leq n+1$ there is an interpretation *I* for which $Matche(e_e^i, e_a^i, I)$ and $Sat(q_i, I) = \text{true}$ (again from the definition of *Matche*, *Matchs*, *Interpretation I*, $Sat(q, I)$ and $Sat(q, e_c, B)$). Therefere, by Fact 3, $e_a^i(k_i) \in Mi$, $i=1, \dots, n+1$. So $M_{n+1} \neq \emptyset$. Since (*) holds for *n*, thus $Next_E(Bn) = Mn$ and, by Fact 1, $Valid_E(Bn) = \text{true}$, therefore according to the definition of semantics of *EBE*s we have $Valid_E(B_{n+1}) = \text{true}$ which implies that $Next_I(B_{n+1})$ is defined and is $\vec{M}_{n+1}$.

VI. Reduction of EBEs

Now we give some rules for reducing *EBE*s.

Statement 2. Let $E1$, $E2$ and $E3$ be *EBE*s. Then

(1) $E1+E1 \approx E1$

(2) $E1+E2 \approx E2+E1$

(3) $(E1+E2)+E3 \approx E1+(E2+E3)$

(4) $(E1;E2);\ E3 \approx E1;\ (E2;E3)$

(5) $E1;(E2+E3) \approx E1;\ E2+E1;\ E3$

(6) $(E1+E2);\ E3 \approx E1;\ E3+E2;\ E3$

(7) $E1\Delta E2 \approx E2\Delta E1$

(8) $(E1\Delta E2)\Delta E3 \approx E1\Delta(E2\Delta E3)$

(9) $E1\Delta(E2+E3) \approx E1\Delta E2+E1\Delta E3$

(10) $$\mathop{\Delta}_{i=1}^{n} p_i[q_i] \approx \sum_{\substack{i,\dots i_n \text{ is} \\ \text{permutation} \\ \text{of } \{1,\dots,n\}}} (p_{i_1}[q_{i_1}];\ \dots;\ p_{i_n}[q_{i_n}])$$

where "$\approx$" means semantical equivalence. This is followed from Theorem 2.

Similarly to *EBE*s we also define the syntactical and semantical equivalence of Implementions.

Definition. Two Implementations $I(M)$ and $I'(M')$ are *syntactically equivalent* if $L(M)=L(M')$, and *semantically equivalent* if $\mathbf{B}(I)=\mathbf{B}(I')$.

Definition. An Implementation $I(M)$ is *minimal* if the automaton M has a minimum number of states.

Theorem 5. There exists an algorithm by means of which we can transform any Implementation $I(M)$ to a minimal Implementation $I'(M')$ so that $I(M)$ and $I'(M')$ are semantically equivalent.

Proof. It is known that there is an algorithm by means of which we can reduce any automaton M to a minimal automaton M' such that $L(M)=L(M')$. The semantical equivalence of $I(M)$ and $I'(M')$ is then followed from the following statement.

Statement 3. If $I(M)$ and $I'(M')$ are Implementations such that $L(M)\subset L(M')$, and for any $u\in L(M')\setminus L(M)$ there are $v\in L(M)$ and w for which $v=uw$, then $I(M)$ and $I'(M')$ are semantically equivalent.

Proof. Let $M=(\Sigma, St, s_0, \delta, F)$ and $M'=(\Sigma', St', s_0', \delta', F')$.

By Fact 4 it is sufficient to prove that for any $Bn=e_c^1...e_c^n$ the following holds:

$$(1)\begin{cases} Matchs(cou_i, f_a, f_t, B_{i-1}),\ i=1, ..., n, \text{ and there is a} \\ \text{sequence } \{Hi\}_{i=1}^n \text{ such that} \\ Hi=\{s|\exists s' \exists q(s' \in H_{i-1} \,\&\, q \in Q \,\&\, s=\delta(s', \langle p_i, q\rangle) \,\&\, Sat(q, e_c^i, B_{i-1})=\text{true}\} \neq \emptyset, \\ \text{and } Next_I(Bi)=Hi,\ i=1, ..., n,\ Ho=\{s_0\}. \end{cases}$$

iff

$$(2)\begin{cases} Matchs(cou_i, f_a, f_t, B_{i-1}),\ i=1, ..., n, \text{ and there is a sequence } \{H'i\}_{i=1}^n \\ \text{such that} \\ H_i'=\{s|\exists s' \exists q(s' \in H_{i-1}' \,\&\, q \in Q \,\&\, s=\delta'(s', \langle p_i, q\rangle) \,\&\, Sat(q, e_c^i, B_{i-1})=\text{true}\} \neq \emptyset, \\ \text{and } Next_{I'}(Bi)=Hi,\ i=1, ..., n,\ H'o=\{s_0'\}. \end{cases}$$

This is proved by induction on n.

1) It is easy to see that the statement holds for $n=1$.

2) Assume that the statement holds for n, we prove that it holds for $n+1$, too.

(1)⇒(2). Suppose that (1) holds for $n+1$. Then (2) holds for n. We have yet to prove that $H_{n+1}'\neq\emptyset$ and $Next_{I'}(B_{n+1})=H_{n+1}'$. By Fact 5 we have the sequences $\{s_i\}_{i=1}^{n+1}$, and $\{e^i\}_{i=1}^{n+1}$ for which $s_i\in Hi$, $s_i=\delta(s_{i-1}, e^i)$, $Sat(q_i, e_c^i, B_{i-1})=\text{true}$, $e^i= =\langle p_i, q_i\rangle$, $i=1, ..., n+1$. Then according to Restriction there is a u for which $e'...e^{n+1}u\in L(M)$. Since $L(M)\subset L(M')$ thus there is a sequence $\{s_i'\}_{i=0}^{n+1}$ for which $s_i'=\delta'(s_{i-1}', e^i)$. So by Fact 6 $s_i'\in H'i$, $i=0, ..., n+1$, that is $H_{n+1}'\neq\emptyset$. Since (2) holds for n, thus $Next_{I'}(Bn)=H'n$ and, by Fact 4, $Valid_{I'}(Bn)=\text{true}$, so $Valid_{I'}(B_{n+1})=$ true (according to the definition of the semantics of Implementation) which implies that $Next_{I'}(B_{n+1})$ is defined and is H_{n+1}'.

(2)⇒(1). Suppose that (2) holds for $n+1$. Then (1) holds for n. We have yet to prove that $H_{n+1}\neq\emptyset$ and $Next_I(B_{n+1})=H_{n+1}$.

By Fact 5 we have the sequences $\{s_i\}_{i=0}^{n+1}$ and $\{e_i\}_{i=1}^{n+1}$ for which $s_i\in H'i$, $s_i=\delta'(s_i, e^i)$, $Sat(q_i, e_c^i, B_{i-1})=\text{true}$, $e^i=\langle p_i, q_i\rangle$, $i=1, ..., n+1$. Then according to Restriction there is a u for which $e^1...e^{n+1}u\in L(M')$. We have two cases:

$$\text{either}\quad e^1 ... e^{n+1}u\in L(M)$$
$$\text{or}\quad e^1 ... e^{n+1}u\in L(M')\setminus L(M).$$

In the second case there is u' for which $e^1...e^{n+1}uu'\in L(M)$. So in both cases, we have the sequence $\{s_i'\}_{i=0}^{n+1}$ for which $s_i'=\delta(s_{i=1}', e^i)$, $i=1, ..., n+1$. Therefore, by Fact 6, $s_i'\in Hi$, $i=0, 1, ..., n+1$, that is $H_{n+1}\neq\emptyset$. Now by the same argument seen above we get $Next_I(B_{n+1})=H_{n+1}$.

Abstract

A language, called *EBE*, for specifying the expected behavior of programs during debugging is presented. *EBE* is an extended version of *GPE* (Generalized Path Expressions) [1] with the operator shuffle. The syntax and semantics of *EBE* is formally defined. Some properties of *EBE*s are discussed. Then an implementation of *EBE* is presented. Correctness of implementation is also proved.

DEPARTMENT OF COMPUTER SCIENCE
EÖTVÖS LORÁND UNIVERSITY
MÚZEUM KRT. 6—8
BUDAPEST, HUNGARY
H—1088

References

[1] BRUEGGE, B. and HIBBARD, P., Generalized Path Expressions: A High-Level Debugging Mechanism, The J. of Sys. and Soft. 3, 256—276 (1983).
[2] LAVENTHAL, M. S., Synthesis of synchronization code for data abstractions, M.I.T. Laboratory for Comp. Sci., 1978.
[3] VARGA, L., Rendszerprogramok elmélete és gyakorlata. Akadémiai Kiadó, Budapest 1978 (in Hungarian).
[4] GÉCSEG, F. and PEÁK, I., Algebraic Theory of Automata, Akadémiai Kiadó, Budapest, 1972.
[5] NGUYEN HUU CHIEN, Sequential program debugging with Path Expressions, conference on Automata, Languages and Programming Systems, Salgótarján, May 1986 (Hungary).
[6] NGUYEN HUU CHIEN, EBE: A Language for Specifying the Expected Behavior of Programs in Debugging, 2nd conference of Program Designers, Budapest, L. Eötvös University, July, 1986.

(Received Sept. 3, 1986)