вать функциональную безопасность и, как следствие, стабильность информационных систем, используемых в критичных приложениях.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Симанков В.С., Сундеев П.В.* Системный анализ функциональной стабильности критичных информационных систем. Монография / Под науч. ред. *В.С. Симанкова.* - Краснодар. 2004. - 204 с.

2. *Qrenander U.* Qeneral Pattern Theory, Oxford University Press, 1993. - 904 p.

3. *Шуткин Л. В.* Парадигма модульного мышления в компьютерной науке и практике // М. НТИ, Сер. 2. 2004. № 10. С. 1-12.

В.Н. Хализев
Россия, г. Краснодар, КубГТУ

АЛГЕБРАИЧЕСКАЯ МОДЕЛЬ АНАЛИЗА БЕЗОПАСНОСТИ ПРОГРАММНЫХ СРЕДСТВ АС

В статье рассматривается подход к оценке безопасности программ и алгоритмов систем критических приложений, базирующийся на алгебраической модели описания семантики программ и алгоритмов, основанной на системе алгоритмических алгебр (САА) Глушкова В.М. [1,2], символьном выполнении программ [3], элиминации инвариантов циклов [4].

Алгебра Глушкова модифицирована введением операций семантического умножения и сложения, определяющих поток данных в алгоритме. Введено формальное понятие семантической конструкции, соответствующей интуитивному понятию алгоритмической конструкции или схеме в [2].

Задача контроля соответствия реальных и декларируемых функций программ и алгоритмов формулируется в виде задачи идентификации, как и в [3], но в отличие от G - выполнения, определённого только для некоторых типов регулярных алгоритмов, рассматриваются произвольные типы конструкций и соответствующие подстановки из базы знаний.

Совокупность базовых алгоритмов и проблемных семантических конструкций составляет базу знаний экспертной системы анализа программ. Построение базы знаний может быть проведено как экспертами вручную, так и путем интерактивного обучения базы знаний на примерах корректных программ.

Математическая модель описания программ и алгоритмов

Задачу контроля соответствия реальных и декларируемых функций программы можно интерпретировать как задачу определения семантики (смысла) того, что делает эта программа в терминах и понятиях заказчика.

Процесс семантического анализа или аннотирования программы можно рассматривать как процесс, обратный синтезу программ, по методу многоуровневого структурного проектирования (МСПП) [2], использующему систему алгоритмических алгебр (САА) Глушкова [1]. Запись алгоритмов в САА-схемах естественна, самодокументирована, ориентирована на класс алгоритмов, независима от языка программирования и позволяет в ходе уточнения семантики осуществлять синтез с оптимизацией по выбранным критериям, т. е. осуществлять трансформационный синтез программ. В этом смысле подход адекватен объектно-ориентированному программированию (ООП).

Для решения задачи анализа, т. е. определения принадлежности какой-либо САА-схемы к классу алгоритмов, необходима формализация понятия класса алгоритма путем расширения понятия САА-схемы до более общего понятия - семантической схемы алгоритма как "нефиксированной" САА-схемы, определяющей смы-

словое преобразование входной информации в выходную без строгой упорядоченности элементов схемы. Семантическая схема алгоритма (ССА-схема) с заданным отношением нестрогого порядка над элементами инвариантна к перестановкам и эквивалентным преобразованиям любых элементов, не меняющих смысла данной ССА-схемы, а также вводу любого количества несущественных промежуточных преобразований.

Подобным образом задача семантического анализа неоднократно ставилась как перспективная не только для целей доказательства правильности программ [5,6], но и как процедура "понимания" алгоритма на формальном языке в базах знаний интеллектуальных систем, например в системе PROUST, при обучении алгоритмическим языкам [7].

Неформально процесс семантического анализа выглядит следующим образом. Алгоритму (программе) А ставится в соответствие выражение Wa алгебры семантик, которыми представляется класс "семантически эквивалентных" алгоритмов. Выражение Wa путем алгебраических преобразований приводится к нормальной форме, затем итерационным алгоритмом в выражении Wa отыскиваются (идентифицируются) типовые семантические конструкции из базы знаний (БЗ) системы анализа и производятся обратные подстановки аналогично правилам редукции или элиминации циклов. В БЗ содержатся все возможные "верные" семантики, типовые ошибки, а также семантики программных закладок (ПЗ) с типовыми разрушающими программными воздействиями (РПВ).В результате итерационного построения всех возможных семантик, начиная с выражения Wa, получаем совокупность выражений на языке БЗ, описывающих смысл программы А с точки зрения знаний, накопленных в БЗ на данный момент. По сути в БЗ накапливаются спецификации типовых алгоритмов (программ) в данной предметной области, что делает данный подход похожим на подход, описанный в [7], однако существенным отличием описываемого алгоритма идентификации является использование как формально-логических эквивалентных преобразований термов, так и эвристических знаний экспертов о разных способах программирования смысловых конструкций в проблемно-ориентированных областях знаний.

Формализуем понятие семантической схемы (семантики) по аналогии со схемами программ Глушкова (САА-схемами), а также общими принципами моделирования вычислительной среды, применяемыми в теории схем программ.

Множеству элементов памяти вычислительной среды M={m} ставится в соответствие счетное множество S(M)=M, называемое множеством состояний элементов памяти. Семантической конструкцией (семантикой) в памяти M назовём элемент R_i частично определенного отображения вида f: M→M ,отображающего множество S(M) в себя. Элемент α_i частично определённого отображения вида h:M→{0,1} называется условием.

В алгоритмических алгебрах Глушкова элемент R_i назван оператором, ему соответствует оператор r_i языка программирования схем программ.

Совокупность всех элементов памяти, исходное состояние которых влияет на действие семантики R_i, назовём областью входных данных семантики, или D-областью. Совокупность элементов памяти, состояние которых может меняться семантикой R_i, назовём V-областью.

Определим операции семантической суммы и произведения следующим образом. Базовый набор семантик $R=\{R_1,R_2,...,R_n\}$ является однозначным отображением операторов $\{r_1,r_2,...,r_n\}$ языка программирования или САА.

Композиция операторов r_1 и r_2 определяется в зависимости от наличия связи выходных переменных оператора r_1 и входных переменных оператора r_2 либо как

семантическое произведение $R_1 \otimes R_2$, при $V(R_1) \cap D(R_2) \neq 0$, либо как семантическая сумма $R_1 \oplus R_2$ в противном случае.

Существенным отличием операции суммы над семантиками от операции дизъюнкции ($\vee$) в САА является то, что в записи $R_1 \oplus R_2$ выполняются параллельно и R_1 и R_2. Это напоминает операцию асинхронной дизъюнкции ($\vee$) из САА-М, однако сумма ($\oplus$) описывает не параллельный процесс, а результат обычного последовательного выполнения операторов r_1 и r_2.

Операция элементарного ветвления (∇) определяется как

$$\alpha \nabla R = \begin{cases} R \text{ при } \alpha=1; \\ e \text{ при } \alpha=0, \end{cases}$$

где e - пустая семантика.

Оператор дизъюнкции САА связан с семантическими операторами соотношением: $({}_{\alpha}r_1 \vee r_2) = \alpha \nabla R_1 \oplus \neg\alpha \nabla R_2$.

Оператору циклирования САА соответствует циклический оператор над семантиками: $({}_{\alpha}r_1) = \alpha \, \mathbf{O} \, R_1$.

Пусть Ω - сигнатура операций, включающая, кроме булевых операций над условиями, операции семантического сложения, умножения, циклирования и ветвления ($\oplus, \otimes, \mathbf{O}, \nabla$), L - множество логических условий. Тогда двухосновную алгебру $A=<R,L,\Omega>$ можно назвать системой семантических алгебр (ССА) по аналогии с САА.

Пусть w - семантика с пустой областью определения D(w)=0, а e - семантика $S(M) \rightarrow S(M)$, т. е. "единица" алгебры A. Тогда в алгебре семантик A действуют следующие тождественные соотношения:

$$\begin{array}{ll}
\alpha \nabla (R_1 \oplus R_2) = \alpha \nabla R_1 \oplus \alpha \nabla R_2; & \alpha \, \mathbf{O} \, (R_1 \oplus R_2) = \alpha \, \mathbf{O} \, R_1 \oplus \alpha \, \mathbf{O} \, R_2; \\
\alpha \nabla \alpha \nabla R = \alpha \nabla R; & \alpha \nabla \alpha \, \mathbf{O} \, R = \alpha \, \mathbf{O} \, R; \\
\alpha \, \mathbf{O} \, \neg\alpha \nabla R = w; & \alpha \nabla \neg\alpha \, \mathbf{O} \, R = w; \\
R \oplus R = R; & R_1 \otimes (R_2 \oplus R_3) = R_1 \otimes R_2 \oplus R_1 \otimes R_3; \\
R_1 \oplus R_2 = R_2 \oplus R_1; & R \oplus w = w \oplus R = R; \\
R \otimes w = w \otimes R = w; & R \otimes e = e \otimes R = R.
\end{array} \qquad (1)$$

Процедура построения выражения Wa алгебры семантик, соответствующего программе (алгоритму A), формализуется следующим образом.

Для каждой точки x_i алгоритма могут быть составлены уравнения вида

$$\begin{array}{l}
x_1 = W_1[R_1,R_2,\ldots,R_N,x_1,x_2,\ldots,x_n], \\
x_2 = W_2[R_1,R_2,\ldots,R_N,x_1,x_2,\ldots,x_n], \\
\quad\quad . \; . \; . \\
x_k = W_k[R_1,R_2,\ldots,R_N,x_1,x_2,\ldots,x_n],
\end{array} \qquad (2)$$

или в векторной форме:

$$X = W\,[R,X].$$

Решение системы уравнений (2) заключается в нахождении последовательности выражений $x_1,x_2,\ldots,x_n$, таких, что уравнения превращаются в тождества. Решение состоит в итеративном процессе подстановки вместо операторов r_i их семантик R_i, вычислении нормальных форм семантических выражений вида $\Sigma\Pi$ для

линейных участков путем их символьного выполнения (либо G-выполнения) и выполнении остальных подстановок, начиная с самых внутренних вложений:

$r_i \to R_i$; $r_i\ r_j \to R_i$ sign R_j;
if α then $R_i \to \alpha \nabla R_i$; while α do $R_i \to \alpha$ O R_i;

$$\text{sign} = \begin{cases} \otimes, \text{ при } V(R_i) \cap W(R_j) \neq \varnothing; \\ \oplus, \text{ при } V(R_i) \cap W(R_j) = \varnothing. \end{cases}$$

Математическая модель процесса анализа соответствия реальных и декларируемых функций программ и алгоритмов

Суперпозицию операций сигнатуры Ω над базисом $R \cup L = \Sigma$ назовём схемой семантики $W(\Sigma)$. Каждому алгоритму (программе, регулярной схеме) однозначно соответствует её схема семантики. Обратное утверждение неверно.

Схемы $W_1(\Sigma)$ и $W_2(\Sigma)$ эквивалентны, $W_1(\Sigma)=W_2(\Sigma)$, если $K(W_1)=K(W_2)$, где $K(W_1)$, $K(W_2)$ - множества конфигураций, порожденные схемами W_1 и W_2 соответственно при применении эквивалентных преобразований (1).

Алгоритмы A_1 и A_2 являются структурно-эквивалентными, если $W_1(A_1)= W_2(A_2)$, т. е. $K(W_1)=K(W_2)$ и при этом в ходе доказательства участвовали все аксиомы алгебры семантик.

Алгоритмы A_1 и A_2 являются логико-термально эквивалентными, если $K(W_1)=K(W_2)$ и при этом в ходе доказательства участвовали все аксиомы алгебры семантик, а также аксиомы математики входящих в алгоритмы термов.

Если к формальным алгебраическим и математическим аксиомам добавить экспертные знания об эквивалентных алгоритмах, получим систему доказательства смысловой эквивалентности алгоритма или систему анализа смысла алгоритма на основе экспертной базы знаний.

Экспертные знания о построении типовых алгоритмов представляют собой подстановки типа $S = W[R,S]$, где S - денотат аксиомы, выражающий смысл (семантику) преобразования данных, W - концепт, описывающий содержание этого преобразования схемой семантической конструкции.

Иерархическое множество аксиом вида

$$\begin{aligned} &S_{11} = W_{11}[R_1,R_2,\ldots,R_n], \\ &S_{12} = W_{12}[R_1,R_2,\ldots,R_n], \\ &\ldots. \\ &S_{1k} = W_{1k}[R_1,R_2,\ldots,R_n], \end{aligned} \qquad (3)$$

или в векторной форме $S^1 = W^1[R]$, описывают подстановки первого уровня, подстановки вида: $S^2 = W^2[R, S^1]$ – второго уровня, и так далее до некоторого уровня m: $S^m = W^m[R, S^{m-1}]$, и составляют БЗ экспертной системы.

Алгоритмы A_1 и A_2 являются семантически эквивалентными, если $K(W_1)=K(W_2)$ и при этом в ходе доказательства участвовали все аксиомы алгебры семантик, аксиомы математики входящих в алгоритмы термов, а также аксиомы экспертных знаний (3).

Процесс семантического анализа программы, представленной виде алгебраического выражения Wa, для контроля соответствия реальных и декларируемых функций программы, а также наличия ПЗ, подробно описан на примерах в [6]. Он состоит в нахождении предельного решения системы алгебраических уравнений

K(X)=K(Wa), составляющих запрос к экспертной системе, путем итерационного применения подстановок из базы знаний системы.

Численное значение уровня безопасности определяется прямым измерением отношения количества идентифицированных семантик в программе ко всему количеству семантик используемой модели (при отсутствии семантик прямо опознанных как программные закладки).

Совокупность базовых алгоритмов и проблемных семантических конструкций составляют базу знаний экспертной системы анализа программ. Построение базы знаний может быть проведено как экспертами вручную , так и путём интерактивного обучения базы знаний на примерах известных разрушающих программных средств и других средств информационного нападения.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Глушков В.М.* Теория автоматов и формальное преобразование программ// Кибернетика. 1965. №5. С.1-10.

2. Многоуровневое структурное проектирование программ: Теоретические основы, инструментарий/ Е.Л.Ющенко, Г.Е.Цейтлин и др. - М.: Финансы и статистика, 1989. -268с.

3. *Костырко В.С.,* Банулин А.В. Об индуктивном синтезе инвариантных утверждений и функций программ // Кибернетика.-1986.-№1.

4. *Непомнящий В.А., Рякин О.М.* Прикладные методы верификации программ./ Под ред. Ершова А.П.- М.: Радио и связь, 1988. - 256с.

5. *Capes Jones T.* Reusability in Programming: A Survey of the State of the Art.// IEEE Transactions on Software Engineering, September 1984. P.488 - 494.

6. *Хализев В.Н., Марков В.Н.* Подход к верификации программ. // Программные продукты и системы. 1993. №2. С.36-39.

7. *Льюис Д., Эллиот С.* Proust (Автоматический отладчик для программ на языке Паскаль).// Реальность и прогнозы искусственного интеллекта / Под ред. *Стефанюка В.Л.* - М.: Мир 1987.