УДК 338.24

Создание самокорректирующихся программ для решения прикладных задач

© О.В. Казарин[1], В.Ю. Скиба[2]

[1] Институт проблем информационной безопасности МГУ им. М.В. Ломоносова, Москва, 119234, Россия
[2] МГТУ им. Н.Э. Баумана, Москва, 105005, Россия

Исследована возможность реализации самокорректирующихся программ для: решения прикладных задач — радиолокации, баллистики и навигации; создания инструментальных библиотек и пакетов прикладных программ; разработки подсистем защиты информации от несанкционированного доступа; надежной отладки программного обеспечения.

Ключевые слова: *проактивная защита, защита программного обеспечения, самотестирующиеся и самокорректирующиеся программы.*

В современном мире наблюдается устойчивая тенденция к увеличению технологического трансфера как между предприятиями в пределах одного государства, так и между различными странами. Технологический трансфер включает не только появление на рынке новых технологий и нового оборудования, но и умение работать с ними. В отраслях промышленности, в которых роль интеллектуальной собственности существенна, таких как научные исследования, создание образцов космической техники, разработка программного обеспечения, доступ к ресурсам и разработкам материнской компании, она приводит к получению выгод, намного превышающих те, которые могли быть получены в результате вливания капитала. Это объясняет интерес многих правительств к тому, чтобы транснациональные корпорации размещали научно-исследовательские центры в их странах.

В этих условиях важность обеспечения информационной безопасности только возрастает. В статье исследована возможность: создания самокорректирующихся программ для радиолокации, баллистики и навигации, инструментальных библиотек и пакетов прикладных программ; разработки подсистем защиты информации от несанкционированного доступа; надежной отладки программного обеспечения.

Реализация таких программ возможна, так как существует множество функций, обладающих свойством рандомизированной алгоритмической самосводимости. Процесс самокоррекции в этом случае независим от используемых при написании защищаемой программы языка программирования и средств автоматизации программирования, что существенно повышает оперативность исследования программ и, сле-

довательно, обнаружения в ней программных ошибок и дефектов деструктивного типа. Пространственно-временную сложность тестирующего модуля для таких программ составляет константный мультипликативный фактор от сложности самой защищаемой программы. Следовательно, весь процесс самотестирования/самокоррекции подобного рода достаточно эффективен.

Рассмотрим *содержание задачи разработки самотестирующихся и самокорректирующихся программ и их сочетаний*.

Пусть необходимо написать программу P для вычисления функции f так, чтобы $P(x) = f(x)$ для всех значений x (все формальные определения самотестирующихся и самокорректирующихся программ можно найти в работах О.В. Казарина, В.Ю. Скибы [3, 5]).

Один из методов решения данной проблемы заключается в создании так называемых самокорректирующихся и самотестирующихся программ, которые позволяют оценить вероятность некорректности результата выполнения программы, т. е. $P(x) \neq f(x)$, и корректно вычислить $f(x)$ для любых x, в случае, если сама программа P на большинстве наборов своих входных данных (но не всех) работает корректно.

Чтобы добиться корректного результата выполнения программы P, вычисляющей функцию f, необходимо написать такую программу T_f, которая позволила бы оценить вероятность того, что $P(x) \neq f(x)$ для любых x. Такая вероятность будет называться *вероятностью ошибки* выполнения программы P. При этом T_f может обращаться к P как к своей подпрограмме. Обязательным условием для программы T_f является ее принципиальное *временно́е отличие* от любой корректной программы вычисления функции f, в том смысле, что время выполнения программы T_f без учета вызовов программы P должно быть значительно меньше, чем время выполнения любой корректной программы для вычисления f. В этом случае вычисления согласно инструкциям T_f некоторым количественным образом должны отличаться от вычислений функции f и самотестирующаяся программа может рассматриваться как независимый шаг при верификации программы P, которая предположительно вычисляет функцию f. Кроме того, желательное свойство для программы T_f должно заключаться в том, чтобы ее код был, насколько это возможно, более простым и отличался бы от кода тестируемой программы для вычисления f. И более того, желательно, чтобы программа T_f была эффективной в том смысле, что время выполнения T_f даже с учетом времени, затраченного на вызовы P, должно составлять лишь константный мультипликативный фактор от времени выполнения P. Таким образом, самотестирование должно лишь незначительно замедлять время выполнения программы P.

Пусть π означает некоторую вычислительную задачу и/или некоторую задачу поиска решения. Для x, рассматриваемого в качестве входа задачи, пусть $\pi(x)$ обозначает результат решения задачи π. Пусть P — программа, предназначенная для решения задачи π. Будем говорить, что P имеет дефект, если для некоторого входа x задачи π имеет место $P(x) \neq \pi(x)$. Определим *программный чекер* C_π для задачи π следующим образом. Чекер $C_\pi^P(x,k)$ является вероятностной программой, удовлетворяющей следующим условиям.

Для любой программы P (предположительно решающей задачу π), выполняемой на всех входах задачи π, для любого входа x и для любого положительного k (параметра безопасности) имеет место:

если программа P не имеет дефектов, т. е. $P(x) = \pi(x)$ для всех входов x задачи π, то $C_\pi^P(x,k)$ выдаст на выходе ответ «Норма» с вероятностью не менее $1 - 1/2^k$;

если программа P имеет дефекты, т. е. $P(x) \neq \pi(x)$ для всех входов x задачи π, тогда $C_\pi^P(x,k)$ выдаст на выходе ответ «Сбой» с вероятностью не менее $1 - 1/2^k$.

Самокорректирующаяся программа — это вероятностная программа C_f, которая помогает программе P скорректировать саму себя, если только P выдает корректный результат с малой вероятностью ошибки, т. е. для любого x, C_f вызывает программу P для корректного вычисления $f(x)$, в то время как сама P обладает малой вероятностью ошибки.

Самотестирующейся/самокорректирующейся программной парой называется пара программ вида (T_f, C_f). Предположим, пользователь применяет программу P, которая вычисляет f и тестирует саму себя при помощи программы T_f. Если P не проходит такие тесты, тогда по любому x, пользователь может вызвать программу C_f, которая в свою очередь вызывает P для корректного вычисления $f(x)$. Даже если программа P вычисляет значение функции f некорректно для отдельной небольшой доли входных значений, ее все равно можно уверенно использовать для корректного вычисления $f(x)$ для любого x. Кроме того, если в будущем удастся написать программу P′ для вычисления той же функции f, то пара (T_f, C_f) может использоваться для самотестирования и самокоррекции P′ без какой-либо модификации этой пары.

Вероятностная программа M является *оракульной программой*, если она может вызывать другую программу во время выполнения M.

Пусть $x \in I_n$ и пусть $c > 1$ — целое число. Свойство *рандомизированной самосводимости* заключается в том, что существует алгоритм A_1, работающий за время пропорциональное $nO(1)$, посредством ко-

торого функция $f(x)$ может быть выражена через вычислимую функцию F от x, a_1, ..., a_c и $f(a_1)$, ..., $f(a_c)$ и алгоритм A_2, работающий за время, пропорциональное $nO(1)$, посредством которого по данному x можно вычислить a_1, ..., a_c, где каждое a_i выбирается случайно в соответствии с распределением D^p.

Перейдем к рассмотрению вычислительных самотестирующихся и самокорректирующихся примитивов.

- *Элементарная арифметика, целочисленная арифметика и арифметика многократной точности.* Известные элементарные арифметические операции: сложения, вычитания, умножения, деления и возведения в степень, исходя из их фундаментальной математической сущности, обладают свойством рандомизированной самосводимости. Следовательно, для них можно построить самотестирующиеся/самокорректирующиеся программные пары. Это можно показать на примере операции сложения.

Пусть необходимо вычислить функцию $f(a + b)$, которая реализуется программой $P(a, b)$. Для этого необходимо: выбрать случайным образом a_1 и b_1 и вычислить $a_2 = a - a_1$ и $b_2 = b - b_1$. Получение значения функции $f(a + b) = f((a_1 + a_2) + (b_1 + b_2))$ следует из корректного выполнения оракульной программы (T_f, C_f) с 2-кратным вызовом программы $P(a, b)$, т. е. $P(a_1, a_2) + P(b_1, b_2)$. Алгоритм вычисления $f(a + b)$ выполняется при декомпозиции слагаемых на два компонента. Однако, декомпозиция a и b при вычислении функции $f(a + b)$ может осуществляться и на большее число компонентов, что приведет к гораздо большему количеству вызовов программы $P(a, b)$, но в то же время позволит значительно снизить вероятность ошибки при выполнении программы $P(a, b)$. Аналогично строятся самотестирующиеся/самокорректирующиеся программные пары для операций умножения и возведения в степень, вычитания и деления, а также операций целочисленной арифметики и арифметики многократной точности [3].

- *Тригонометрические и гиперболические функции.* Пусть необходимо вычислить тригонометрическую функцию $\sin\gamma$.

Выберем случайным образом угол и вычислим угол такой, что $\beta = \gamma - \alpha$.

Из формул сложения для тригонометрических функций известно, что $\sin(\alpha + \beta) = \sin\alpha\cos\beta + \cos\alpha\sin\beta$, а исходя из основного тригонометрического тождества $\sin^2\beta + \cos^2\beta = 1$, можно получить $\cos\beta = \sqrt{1 - \sin^2\beta}$. Тогда $\sin(\alpha + \beta) = \sin\alpha\sqrt{1 - \sin^2\beta} + \sqrt{1 - \sin^2\alpha}\sin\beta$.

Таким образом, чтобы получить самотестирующуюся программу для вычисления функции $\sin\gamma$, необходимо вызвать 6 раз программу вычисления синуса для углов α и β, где $\gamma = \alpha + \beta$.

Аналогичным образом строятся такие же пары и для формулы вычитания для синуса:

$$\sin(\alpha-\beta)=\sin\alpha\sqrt{1-\sin^2\beta}-\sqrt{1-\sin^2\alpha}\sin\beta,$$

а также для формул сложения и вычитания для косинуса, тангенса и котангенса:

$$\cos(\alpha+\beta)=\cos\alpha\cos\beta-\sqrt{1-\cos^2\alpha}\sqrt{1-\cos^2\beta};$$

$$\cos(\alpha-\beta)=\cos\alpha\cos\beta+\sqrt{1-\cos^2\alpha}\sqrt{1-\cos^2\beta};$$

$$\operatorname{tg}(\alpha+\beta)=\frac{\operatorname{tg}\alpha+\operatorname{tg}\beta}{1-\operatorname{tg}\alpha\operatorname{tg}\beta};$$

$$\operatorname{tg}(\alpha-\beta)=\frac{\operatorname{tg}\alpha-\operatorname{tg}\beta}{1+\operatorname{tg}\alpha\operatorname{tg}\beta};$$

где

$$\alpha\neq\frac{\pi}{2}+\pi n,\quad \beta\neq\frac{\pi}{2}+\pi n$$

и, соответственно,

$$\alpha+\beta\neq\frac{\pi}{2}+\pi n,\ \ \alpha-\beta\neq\frac{\pi}{2}+\pi n$$

и

$$\operatorname{ctg}(\alpha+\beta)=\frac{\operatorname{ctg}\alpha\operatorname{ctg}\beta-1}{\operatorname{ctg}\beta+\operatorname{ctg}\alpha};$$

$$\operatorname{ctg}(\alpha-\beta)=\frac{\operatorname{ctg}\alpha\operatorname{ctg}\beta+1}{\operatorname{ctg}\beta-\operatorname{ctg}\alpha},$$

где

$$\alpha\neq\pi n,\ \ \beta\neq\pi n$$

и, соответственно,

$$\alpha+\beta\neq\pi n,\ \alpha-\beta\neq\pi n,\ \ n\in Z.$$

С точки зрения формул сложения и вычитания гиперболические функции обладают теми же свойствами, что и соответствующие тригонометрические. Например:

$$sh(\alpha\pm\beta)=sh(\alpha)\,ch(\beta)\pm ch(\alpha)\,sh(\beta);$$

$$ch(\alpha\pm\beta)=ch(\alpha)\,ch(\beta)\pm sh(\alpha)\,sh(\beta).$$

Кроме того, имеют место следующие тождества для гиперболических функций, которые могут использоваться при создании самотестирующихся/самокорректирующихся программных пар:

$$sh(2\alpha) = 2sh(\alpha)ch(\alpha);$$

$$ch(2\alpha) = ch^2(\alpha) + sh^2(\alpha);$$

$$ch(0) = 1.$$

Исходя из основного тождества гиперболической геометрии: $ch^2(\alpha) - sh^2(\alpha) = 1$ можно получить аналогичные вышеприведенным самотестирующиеся/самокорректирующиеся программы для вычисления гиперболических синуса, косинуса, тангенса и котангенса.

- *Вычисления над матрицами.* Автором одной из первых работ в области вероятностных алгоритмов, написанной еще в 1979 г., которая стала основополагающей в области методологии самотестирования, был Р. Фрейвалдс. Он предложил простой и элегантный чекер для задачи умножения матриц [11]. Используя чекер Фрейвалдса, можно достаточно просто построить самотестирующуюся/самокорректирующуюся программную пару для умножения матриц [9]. Аналогичным образом строятся самотестирующиеся/самокорректирующиеся программные пары для других операций над матрицами: определения детерминанта, инверсии и транспонирования матриц, определения ранга матрицы [3].

- *Аппроксимирующие функции.* Пусть для данной функции f и границы ошибки δ, входа x программа $P(x)$, вычисляющая функцию f, приблизительно корректна, если $|P(x) - f(x)| \le \delta$. Будем также считать, что $P(\delta, \varepsilon)$ аппроксимирует функцию f на области D, если $|P - f| \le \delta$ на более чем $1 - \varepsilon$ элементах области D. Исходя из работ [12, 13] можно показать, как выполнить аппроксимирующие проверку, самотестирование и самокоррекцию полиномов и функциональных уравнений.

Кроме того, в ряде работ показано, как строить аппроксимирующие чекеры для ряда модулярных и логарифмических функций, а также для умножения и инверсии матриц, решения систем линейных уравнений и определения детерминантов. Также исследованы задачи аппроксимирующего самотестирования операций деления с плавающей точкой, одномерных полиномов степени до 9 включительно и многомерных полиномов, а также для тригонометрических и гиперболических функций [3].

- *Другие самотестирующиеся/самокорректирующиеся примитивы.* В работе [3] приведен большой набор чекеров самотестирующихся/самокорректирующихся программ для:

- решения теоретико-групповых и теоретико-числовых задач;
- вычислений над векторами и полиномами;
- вычислений линейных рекуррентных соотношений вида

$$f(n) = \sum_{i=1}^{d} c_i f(n-i);$$

- генерирования псевдослучайных чисел;
- разработки параллельных программ и построения схемы константной глубины для проверки мажоритарных функций; и т.п.

Предварительный анализ показывает, что все перечисленные в этом и предыдущих подразделах самотестирующиеся/самокорректирующиеся примитивы имеют большой прикладной потенциал в различных областях вычислительной математики. В следующем разделе показано, как некоторые из них могут использоваться для решения задач самокоррекции в конкретных практических задачах.

Далее проанализируем использование самокорректирующихся программ при решении задач радиолокации, баллистики и навигации.

В области *радиолокации* необходимо решать математические задачи, связанные с защищенностью от естественных и преднамеренных помех, а также селекцией движущихся целей — обнаружения отраженных целями сигналов, маскируемых радиоволнами, отраженными от местных предметов — зданий, холмов, леса (при наблюдении низколетящих самолетов и снарядов или объектов, движущихся по земле), либо от волнующегося моря (при наблюдении перископов подводных лодок), либо от «облака» пассивных дипольных помех (при наблюдении воздушных объектов) и т.д.

Так, например, в радиолокации применяют цилиндрическую систему с координатами: горизонтальная дальность D_h, азимут α и высота H. Горизонтальная дальность $D_h = OM'$ является проекцией линии наклонной дальности OM на горизонтальную плоскость. Высота цели H равна длине перпендикуляра, опущенного из M на горизонтальную плоскость, т. е. $H = MM'$. Понятно, что $D_h = D\cos\beta$; $H = D\sin\beta$; $D = \sqrt{D_h^2 + H^2}$, где D — наклонная дальность, которая является расстоянием по прямой от радиолокационной станции до цели.

Ранее уже было указано, что при вычислении значений некоторых из этих функций, в том числе аппроксимирующим образом, можно применить самотестирующиеся/самокорректирующиеся процедуры.

В области *баллистических задач* рассмотрим траекторию полета летательного аппарата, движущегося в поле тяготения Земли, например, траекторию полета тактических и оперативно-тактических (не межконтинентальных стратегических) ракет. Для таких ракет траекторией полета после активного участка при их пуске является балли-

стическая кривая. Если считать поле тяготения однородным (что справедливо при начальной скорости тела, значительно меньшей первой космической) и пренебречь сопротивлением воздуха, то в первом приближении баллистической кривой будет парабола.

Пусть тело брошено с поверхности Земли с начальной скоростью v_0 под углом α к горизонту. Уравнение траектории движения тела представляет собой параболу

$$y = -\frac{g}{2v_0^2 \cos^2 a} x^2 + \operatorname{tg} a x.$$

Координаты вершины параболы:

$$x_0 = -\frac{\operatorname{tg} a}{-\frac{g}{v_0^2 \cos^2 a}} = v_0^2 \sin a \cos a;$$

$$y_0 = -\frac{v_0^2}{2g} \sin a.$$

Таким образом, дальность полета тела по горизонтали будет

$$l = 2x_0 = -\frac{2v_0^2 \sin a \cos a}{g}.$$

Наибольшая высота подъема тела:

$$y_0 = -\frac{v_0^2}{2g} \sin^2 a.$$

В момент падения тела $y = 0$, $x = l$, а наибольшее значение дальности полета будет

$$l_m = \frac{v_0^2}{g}.$$

Это достигается при $\operatorname{tg}\alpha = \frac{v_0^2}{g t_m} = 1$, т. е. при $\alpha = 45°$. Делая элементарные тригонометрические преобразования, можно найти координаты поражаемой области для попадания ракеты (боеголовки (-ок) ракеты):

$$y \le \frac{v_0^2}{2g} - \frac{g}{2v_0^2} x^2.$$

Для вычисления параметров баллистической кривой требуется выполнение операций сложения, вычитания, умножения, деления и возведения в квадрат, а также вычисление значений функций синуса, косинуса и тангенса. Для всех этих операций можно построить самотестирующиеся/самокорректирующиеся программные пары, что дает возможность посредством композиции самокорректирующихся вычислительных примитивов обеспечить правильное (корректное) получение значений уравнений даже при наличии в программах преднамеренных дефектов, которые вводят (подменяют) некоторые параметры вычислений, например начальную скорость v_0 и угол α.

- *Навигационные задачи*. В системах спутниковой навигации местоположение объекта может определяться методом гиперболической триангуляции, как и в системах сотовой связи при определении местоположения мобильной станции [6]. Для определения объекта на поверхности требуется наличие трех и более базовых станций (чем их больше, тем выше точность измерений и вычисления точки пересечений соответствующих гипербол распространения радиосигнала), а для определения объекта в пространстве аналогичная задача решается для гиперболоидов. Соответственно, вычислительные примитивы для гиперболических функций позволяют строить самотестирующиеся/самокорректирующиеся программные пары для определения местоположения объектов в двухмерном и трехмерном пространстве.

Кроме того, для повышения точности измерений может использоваться способ минимизации функционала посредством безусловного нелинейного метода наименьших квадратов, реализованный через алгоритм наименьшего спуска [6]. Рассмотрим метод наименьших квадратов применительно к задачам двух типов: детерминированной и стохастической с известными вторыми моментами. Детерминированная задача состоит в выборе вектора x таким образом, чтобы минимизировать значение ε_1^2 в уравнении:

$$\varepsilon_1^2 = ||Ax - y|| 2 = x^T A^T Ax - 2y^T Ax + ||y|| 2,$$

где A — известная матрица, y — известный вектор, индекс T обозначает транспонирование матрицы или вектора.

Стохастическая задача наименьших квадратов состоит в выборе детерминированного весового вектора w для минимизации средней квадратической ошибки ε_2^2:

$$\varepsilon_2^2 = E\left(w_z^T - d\right)2 = w_z^T R_z w - 2R_{dz}^T w + E(d_2),$$

где z — случайный вектор; d — случайная величина.

Скалярная случайная величина w_z^T представляет собой линейную комбинацию компонентов случайного вектора z и используется как линейная оценка случайной величины d. Математическое ожидание обозначается символом E, через R_z обозначена матрица вторых моментов случайного вектора z: $R_{dz} = E(dz)$. Аналогично вектор R_{dz} определяется как $R_{dz} = E(dz)$. В работе [8] показана вычислительная эквивалентность стохастической задачи наименьших квадратов с оценкой вторых моментов и соответствующей детерминированной задачи. Поэтому для наших целей достаточен анализ вычислительных методов только для детерминированной задачи наименьших квадратов.

Из вышеприведенных формул видно, что задача наименьших квадратов в итоге сводится к умножению вектора на вектор, умножению матрицы на вектор, сложению и умножению матриц, транспонированию матриц, нахождению математического ожидания и нахождению нормы вектора. Исходя из приведенных результатов, можно констатировать, что для большинства данных функций существуют программные чекеры и самотестирующиеся/самокорректирующиеся программные пары. Следовательно, существует принципиальная возможность последовательной либо параллельной организации работы программных чекеров, самотестирующихся и самокорректирующихся программ для решения задачи наименьших квадратов.

Рассмотрим использование самокорректирующихся программ для создания инструментальных библиотек и пакетов прикладных программ.

В качестве примера приведем создание набора самокорректирующихся программ для инструментальной библиотеки CrypTool [5]. Она предназначена для изучения криптографических примитивов, схем и протоколов, а также для анализа их безопасности. При помощи библиотеки можно разрабатывать программное обеспечение, решающее следующие задачи защиты информации:

- генерация и распределение ключей защиты данных;
- шифрование/дешифрование файлов и сообщений;
- электронная цифровая подпись файлов и сообщений;
- обнаружение ошибок в передаваемых сообщениях;
- аутентификация пользователей и сообщений; и др.

Библиотека функций CrypTool представляет собой набор специализированных функций и типов данных для целочисленной арифметики с операндами многократной точности, модулярных операций, высокоуровневых теоретико-числовых функций (подобных расширенному алгоритму Евклида, псевдопростым вероятностным тестам и т. п.), реализаций режимов симметричных криптосистем (режимов алгоритмов ГОСТ 28147–89, DES и собственной разработки), контрольного суммирования сообщений и файлов, для реализации крип-

тографически стойкой функции хэширования сообщений согласно рекомендации МККТТ X.509, а также электронной цифровой подписи в соответствии со стандартами ГОСТ Р 34.10–94, 10–2001 и схемами RSA, Эль-Гамаля, Шнорра и Sandia.

В состав библиотеки входят типы данных и функции, которые условно можно подразделить на следующие группы.

1. Вспомогательные типы данных по обработке ошибок.

2. Функции и типы данных для реализации алгоритмов симметричного шифрования и контрольного суммирования.

3. Базовый класс чисел многократной точности и функции для реализации арифметических, логических, модулярных и высокоуровневых операций с ними.

4. Классы данных и функции для реализации цифровой подписи в соответствии с алгоритмом RSA.

5. Класс данных и функции для реализации цифровой подписи в соответствии с алгоритмами Эль-Гамаля, Шнорра, DSS, Sandia, ГОСТ Р 34.10–94 и ГОСТ Р 34.10–2001.

Для всех перечисленных теоретико-числовых функций были разработаны самотестирующиеся/самокорректирующиеся программы, которые позволяют, с одной стороны, осуществить эффективную отладку программных реализаций этих функций, с другой — защищать разрабатываемые прикладные программы от потенциальных программных дефектов деструктивного типа, при условии, что на большинстве своих входов (но не всех) эти программы работают корректно.

Как самокорректирующиеся программы используются для подсистем защиты информации от несанкционированного доступа?

Поскольку методы защиты информации используются, в том числе, для обеспечения конфиденциальности и целостности данных, естественно, что программно-техническая реализация этих методов должна быть свободна от программных дефектов деструктивного типа. Таким образом, средства самотестирования и самокоррекции программ могут применяться в современных системах защиты информации от несанкционированного доступа. В работе О.В. Казарина [3] приведены самокорректирующиеся программы для реализации следующих схем:

- системы открытого распределения ключей Диффи — Хеллмана;
- интерактивных систем доказательств и интерактивных систем доказательств с нулевым разглашением;
- криптосистемы (и схемы электронной цифровой подписи) RSA.

Аналогичные программы можно построить для многих других схем защиты, где используются функции, реализующие различные теоретико-числовые операции и модулярную арифметику с операндами многократной точности.

Рассмотрим, как самотестирующиеся и самокорректирующиеся программы используются для надежной отладки программного обеспечения.

Поскольку безошибочное программирование почти невозможно, а ручная отладка для современных программных комплексов почти немыслима, необходимы средства поиска ошибок и их исправления. В каждой современной системе программирования существуют специальные средства отладки программ — *отладчики*, позволяющие в режиме интерпретации установить контрольные точки, выполнить отдельные участки программы и посмотреть результаты работы операторов. В то же время, использование отладчиков в достаточной степени ограничено, так как легко потеряться в деталях сложных структур данных и путей исполнения программы. Пошаговый проход по программе менее продуктивен, чем использование программного кода, который проверяет сам себя в критических точках.

Самотестирующиеся / самокорректирующиеся процедуры, впрочем, как и многие типы простых чекеров, расставленные в критических точках программы, в случае получения некорректного результата сравнения (следовательно, и в случае наличия программных ошибок, как и программных дефектов деструктивного типа (тех из них, которые влияют на правильность результата вычислений)) могут записывать в специальный файл, который должен периодически просматриваться разработчиками данной программы, различные атрибуты срабатывания, либо могут выдавать сигналы предупреждения (на периферийные устройства — мониторы, принтеры, динамики и пр.) в случае появления программной ошибки.

Для повышения надежности отладки разработчиков программных комплексов можно разделить на две группы:

пишущих целевые программы;

2) пишущих самотестирующиеся/самокорректирующиеся программные пары для первых.

Вторая группа должна быть наиболее надежной в организациях — разработчиках программного обеспечения. Исходя из оракульных свойств самотестирующихся/самокорректирующихся программ, очевидно, что разработчикам программных пар необязательно знать внутреннюю структуру целевой программы и ее характеристики (например, управляющий граф программы, глобальные и локальные переменные, внутренние подпрограммы, процедуры и функции). Достаточно того, чтобы на вход их оракульной программы подавались входные переменные, которые должны подаваться на вход целевой программы. Важно отметить, что один раз написанные чекер, самотестирующаяся/ самокорректирующаяся программная пара для конкретной вычислительной задачи будут верны для той же задачи, написанной другим способом (например, на другом языке программирования или с использованием другого вычислительного метода/алгоритма). Это, как правило, будет верно и для программ, которые были модифицированы во время их эксплуатации, при условии, что расставленные контрольные точки (самотестирующиеся процедуры) были оставлены в теле измененной программы.

В данной работе показана принципиальная возможность реализации самокорректирующихся программ в различных предметных областях. Такие программы вместе с самокорректирующимися схемами [7] составляют так называемые самокорректирующиеся среды [2, 5], которые могут стать базовой основой для разработки безопасного программного обеспечения автоматических систем, а также систем, функционирующих в реальном масштабе времени, работающих автономно, и т.п.

Идея самокоррекции программ наряду с другими заложена в парадигму проактивной безопасности компьютерных систем, которую предлагается рассматривать как сравнительно новую концептуальную схему для создания продуктов информационных технологий с высоким оценочным уровнем доверия [1, 2, 4, 5].

ЛИТЕРАТУРА

[1] Казарин О.В. Проактивная безопасность вычислительных систем. *Математика и безопасность информационных технологий. Материалы конференции в МГУ — МаБИТ –2004*. Москва, МЦНМО, 2005, с. 306–320.

[2] Казарин О.В. Проактивная безопасность и самокорректирующиеся среды. *Математика и безопасность информационных технологий. Материалы конференции в МГУ — МаБИТ –2005*. Москва, МЦНМО, 2006, с. 322–334.

[3] Казарин О.В. *Методология защиты программного обеспечения.* Москва, МЦНМО, 2009, 464 с.

[4] Казарин О.В., Скиба В.Ю. Парадигма проактивной безопасности компьютерных систем. *Защита информации. INSIDE*, 2009, № 5, с. 2–9; № 6, с. 2–7.

[5] Казарин О.В., Скиба В.Ю. Применение самокорректирующихся сред для обеспечения проактивной безопасности компьютерных систем. *Известия вузов. Приборостроение,* 2010, № 1, с. 34–39.

[6] Камалов Ю.Б., Служивый М.Н. Определение местоположения мобильного объекта. *Известия Самарского научного центра Российской академии наук,* 2009, № 3 (2), с. 361–368.

[7] Редькин Н.П. Асимптотически минимальные самокорректирующиеся схемы для одной последовательности булевых функций. *Дискретный анализ и исследование операций. Серия 1,* 1996, № 2, с. 62–79.

[8] Уайтхаус Х., Спейзер Дж., Бромли К. Применение параллельных матричных процессоров для обработки сигналов. *Сверхбольшие интегральные схемы и современная обработка сигналов.* Москва, Радио и связь, 1989.

[9] Blum M., Luby M., Rubinfeld R. Self-testing/ correcting with applications to numerical problems. *Proc 22th ACM Symposium on Theory of Computing,* 1990, pp. 73–83.

[10] Ergun F., Kumar S.R., Rubinfield R. Approximate checking of polynomials and functional equations. *Proc. of the 37-th IEEE Symp. on Foundations of Computer Science,* FOCS, 1996, pp. 592–601.

[11] Freivalds R. Fast probabilistic algorithms. *Lecture Notes in Computer Science. Mathematical Foundations of CS,* 1979, v. 74, pp. 57–69.

[12] Gemmel P., Lipton R., Rubinfield R., Sudan M., Wigderson A. Self-testing/correcting for polynomials and for approximate functions. *Proc. 23-rd ACM Symposium on Theory of Computing,* STOC, 1991, pp. 32–42.

[13] Rubinfield R., Sudan M. Robust characterization of polynomials and their applications to program testing. *SIAM J. of Computing,* 1996, № 25 (2), pp. 252–271.

Статья поступила в редакцию 26.07.2013

Ссылку на эту статью просим оформлять следующим образом:

Казарин О.В., Скиба В.Ю. Создание самокорректирующихся программ для решения прикладных задач. *Инженерный журнал: наука и инновации*, *2013, вып. 3*. URL: http://engjournal.ru/catalog/it/asu/653.html

Казарин Олег Викторович — доктор технических наук, старший научный сотрудник, ведущий научный сотрудник Института проблем информационной безопасности МГУ имени М.В. Ломоносова. e-mail: okaz2005@yandex.ru

Скиба Владимир Юрьевич — доктор технических наук, профессор кафедры предпринимательства и внешнеэкономической деятельности МГТУ им. Н.Э. Баумана. Автор свыше 100 научных работ, в том числе 2 монографий и 3 учебных пособий в области разработки систем защиты информации и информационной безопасности предприятий и организаций. e-mail: skiba@eecommission.org