Продолжение таблицы 2

1	2	3	4	5
Метод Якоби				
Δx	– 1,72	– 0,96	0	– 1,377
Δy	+ 1,38	3,23	– 0,175	3,250
Δz	3,05	– 1,46	– 1,220	– 0,170
x	45,29	46,05	30,01	33,632
y	35,39	37,24	34,835	33,260
z	79,06	74,55	98,79	114,84
M	**0,1554**	0,1539	0,01300	0,01498
ψ	0,6220	0,5968	0,98888	0,6980
Метод Ньютона				
Δx	1,82	3,59	1,06	– 0,073
Δy	4,73	8,67	0,11	0,537
Δz	– 2,42	– 5,11	0,07	– 0,055
x	48,83	50,60	31,07	34,937
y	38,74	42,68	35,12	30,547
z	73,59	70,90	100,08	114,955
M	0,15311	0,1528	**0,01303**	0,01529
ψ	0,5887	0,5832	0,9249	0,7040

- самым надежным удобным и простым является метод релаксации, поскольку минимизация функции может осуществляться со сколь угодно малым шагом λ.

СПИСОК ЦИТИРОВАННЫХ ИСТОЧНИКОВ

1. Применение геодезических засечек, их обобщенные схемы и способы машинного решения / П. И. Баран, В. И. Мицкевич, Ю. В. Полищук [и др.]. – М.: Недра, 1986. – 166 с.
2. Химмельблау, Д. М. Прикладное нелинейное программирование / Д. М. Химмельблау; пер. с англ. – М.: Мир, 1975. – 534 с.
3. Вычислительная математика / В. А. Вергасов, И. Г. Журкин, М. В. Красикова [и др.]. – М.: Недра, 1976. – 230 с.
4. Грищенков, Е. В. Оптимизация качества построения геодезических сетей методами нелинейного программирования / Е.В. Грищенков, Л. А. Черкас // Вестник Полоцкого государственного университета. – Сер. Прикладные науки. – 2006. – № 9. – С. 117–120.

Материал поступил в редакцию 17.12.07

GRISCHENKOV E.V., ZUEVA L.F., SINIAKINA N.V. Application of methods of nonlinear programming at designing a ground spatial geodetic label

The methods allowing with the help PC to decide one of tasks of technical designing are considered. The optimum rules of items are defined, for which the borders of area of their arrangement are given, by maintenance of the minimal meanings of criterion functions. It results in the minimal meanings of functions of equal sizes with maximal weight at constant meanings of the standards of measurements.

By development of the specified method of the decision of tasks of optimum designing use elements of the theory of matrixes and some methods of nonlinear programming - relax, Koshi, Jakobi and Newton.

УДК 681.324

Дрогин В.В.

ПРИМЕНЕНИЕ НА ЭТАПЕ КОМПИЛЯЦИИ АВТОМАТИЧЕСКОГО ТЕСТИРОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ НА ПРЕДМЕТ ОБНАРУЖЕНИЯ ОШИБОК

В свете большой стоимости ошибки в коде программного обеспечения, которая может быть причиной уязвимости в программном продукте, а также большой стоимости неавтоматизированного тестирования программного обеспечения, перспективным является внедрение в компилятор возможности обнаружения ошибок в программном обеспечении методом грубой силы (фаззинга). Концепция фаззинга (от англ. fuzzing) совсем недавно привлекла общественное внимание. В 2006 году было обнаружено большое количество ошибок в популярном программном обеспечении, включая и широко распространённый браузер Internet Explorer. Большая часть этих ошибок была обнаружена методом фаззинга. Эффективное применение технологии фаззинга дало толчок к появлению большого количества утилит и техник, появлению первых книг по данной тематике [1]. Большая стоимость и трудоёмкость тестирования программного обеспечения на предмет нахождения в нём уязвимостей, а также заметная эффективность фаззинга ставит законный вопрос о внедрении возможности фаззинга в компилятор.

Безусловно, не все ошибки в программном коде требуют такого подхода. Например, ошибки форматирования строки могут быть определены, используя лишь шаблон. К сожалению, не все уязвимости могут быть описаны так же просто, как ошибки форматирования строки. Определение переполнения буфера, целочисленного переполнения, индексирования массивов или проблем с арифметикой указателей требуют более сложного анализа, чем использование шаблона или приведение типов [2]. Такие уязвимости проявляются в программе, когда некая переменная выходит за некоторый безопасный диапазон. Например, вызов функции strcpy() является уязвимым, когда размер строки источника превосходит размер строки получателя. Определение условий, при которых это может происходить, без исполнения программы является сложной задачей. В любой нетривиальной программе существуют зависимости между данными, которыми манипулирует код, что ещё более усложняет зада-

Дрогин Владислав Валерьевич, *аспирант кафедры безопасности информационных технологий Таганрогского технологического института Южного федерального университета.*

Россия, ТТИ ЮФУ, 347928, г. Таганрог, ул. Чехова, 2.

чу. Анализ движения данных является наиболее подходящей техникой для решения такого рода проблем, она может быть использована в качестве основы системы обнаружения уязвимостей.

Рассмотрим данную технику подробнее. В рамках данной техники применимы следующие понятия.

Определение – левая часть выражения, в котором переменная обновляет значение.

Вычислительное использование – переменная в правой части выражения приравнивания.

Использование пары предикатов – результат появления переменной в булевой форме.

Путь ($i, n_1, n_2, \ldots, n_m, j$) – чётко определённый путь от выхода i до входа j, если от n_1 до n_m не содержится определение X.

При определённом x в узле n_d и вычислительном использовании X в узле $n_{c\text{-}use}$, присутствие чётко определённого пути для X от n_d до $n_{c\text{-}use}$ является ассоциацией определение-вычислительное использование ($n_d, n_{c\text{-}use}, X$).

При определённом x в узле n_d и использовании пары предикатов x в узле $n_{p\text{-}use}$, присутствие чётко определённого пути для x от n_d до $n_{p\text{-}use}$ является парой ассоциация определение-использование пары предикатов (n_d, ($n_{p\text{-}use}$, и), X) и (n_d, ($n_{p\text{-}use}$, л), X) согласно результатам выражения истина и ложь соответственно.

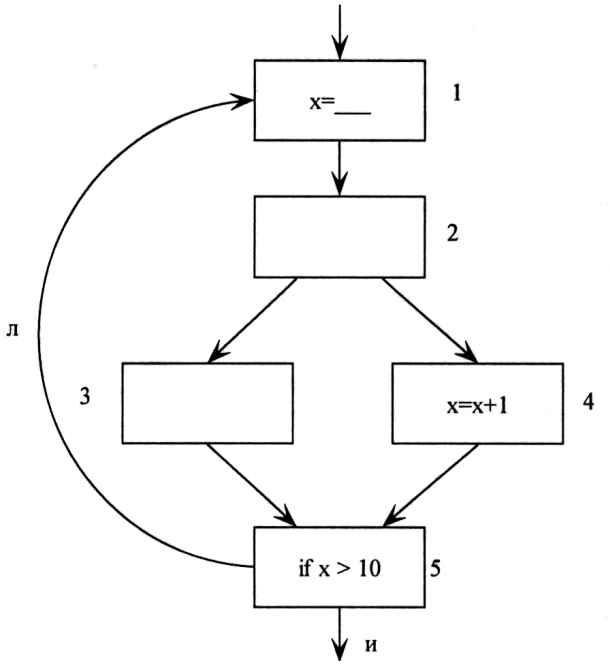


Рис. 1. *Иллюстрация определений*

Пути внутри программы с ассоциациями определение-использование называются ои-путями. Путь ($n_1, \ldots, n_j, n_k$) является ои-путём для переменной X, если n_1 содержит определение X и либо n_k содержит вычислительное использование X и ($n_1, \ldots, n_j, n_k$) чётко определённый простой путь для X (то есть, чётко определённый путь, где все узлы являются определёнными, возможно, за исключением первого и последнего); либо n_j содержит использование пары предикатов $\boldsymbol{X}$ и ($n_1, \ldots, n_j$) является чётко определённым бесцикловым путём для X (то есть, чётко определённым путём, где все узлы являются определёнными).

Пример на рисунке 1 содержит определения X в узлах 1 и 4, одно вычислительное использование X в узле 4 и пару использований пары предикатов X согласно его использованию в узле 5. Пути (1, 2, 3, 5) и (1, 2, 4) являются чётко определёнными путями для X. ОИ-путь (1, 2, 4) является ассоциацией определение-вычислительное использование, тогда как ОИ-ПУТЬ (1, 2, 3, 5) является парой ассоциаций определение-использование пары предикатов (1, (5, и), X) и (1, (5, л), X).

Для данного набора тестовых случаев, пусть P будет полным набором исполненных программой путей для этих тестовых случаев. Для каждого из тестовых критериев покрытия существуют условия, которым должен соответствовать P.

Все определения. Удовлетворяется, если P включает в себя чётко определённый путь от каждого определения до какого-либо соответствующего использования.

Все вычислительные использования. Удовлетворяется, если P включает в себя чётко определённый путь от каждого определения до всех, относящихся к нему, вычислительных использований.

Все вычислительные использования или некоторые использования пар предикатов. Удовлетворяется, если P включает в себя чётко определённый путь от каждого определения до всех соответствующих вычислительных использований. Если определение не имеет вычислительных использований, то P должен включать чётко определённый путь к какому-либо соответствующему использованию пары предикатов.

Все использования пар предикатов. Удовлетворяется, если P включает в себя чётко определённый путь от каждого определения до всех соответствующих использований пар предикатов.

Все использования пар предикатов или некоторые вычислительные использования. Удовлетворяется, если P включает в себя чётко определённый путь от каждого определения до всех соответствующих использований пар предикатов. Если определение не имеет использований пар предикатов, то P должен включать чётко определённый путь к какому-либо соответствующему вычислительному использованию.

Все использования. Удовлетворяется, если P включает в себя чётко определённый путь от каждого определения до каждого его использования, включая вычислительные использования и использования пар предикатов.

Все ОИ-пути. Удовлетворяется, если P включает все ОИ-пути для каждого определения. Следовательно, если существует множество путей между данным определением и использованием, они все должны быть включены.

Все использования ВВ (влияющие на выход). Удовлетворяется, если P включает в себя чётко определённый путь для каждого вычислительного использования и использования пар предикатов. Более того, каждое использование считается протестированным, если оно является влияющим на выход; то есть оно прямо или косвенно влияет на вычисление некоего программного вывода во время работы программы [3].

Из представленных определений различных критериев ясно, что каждый критерий определяет минимальный набор ассоциаций определение-использование, который должен исполняться набором тестовых случаев для удовлетворения требованиям критерия. Однако такой минимальный набор не является уникальным. Так же, как и в случае критериев тестирования управляющей логики, родовидовые отношения существуют и между определёнными критериями тестирования движения данных. На рисунке 2 определены эти родовидовые отношения.

Для полноты картины, в этот рисунок также включены критерии тестирования управляющей логики. Они обозначены пунктирной линией.

Рассмотрим пример на рисунке 3 (см. стр. 48) для иллюстрации критериев тестирования.

На рисунке кроме графа управления даны два тестовых ввода и соответствующие им полные пути. Два полных пути покрывают все узлы и все рёбра в графе управления, и таким образом критериям все-узлы и все-рёбра. Рассмотрим различные критерии покрытия при тестировании движения данных. Таблица, относящаяся к рисунку три, показывает все ассоциации определение-использование, присутствующие в программе. Для каждого критерия движения данных, исключая критерии все-ОИ-пути и все-использования-ВВ, определён минимальный набор ассоциаций. Если данный набор полных путей исполняет определённые ассоциации для критерия теста, то он удовлетворяет критерию теста. В данном примере все определённые критерии теста удовлетворяются двумя полными путями, соответствующими двум вводам программы. Критерии все-ОИ-пути также определяются двумя полными путями, потому что в этом примере только один ОИ-путь соответствует каждой ои-ассоциации.

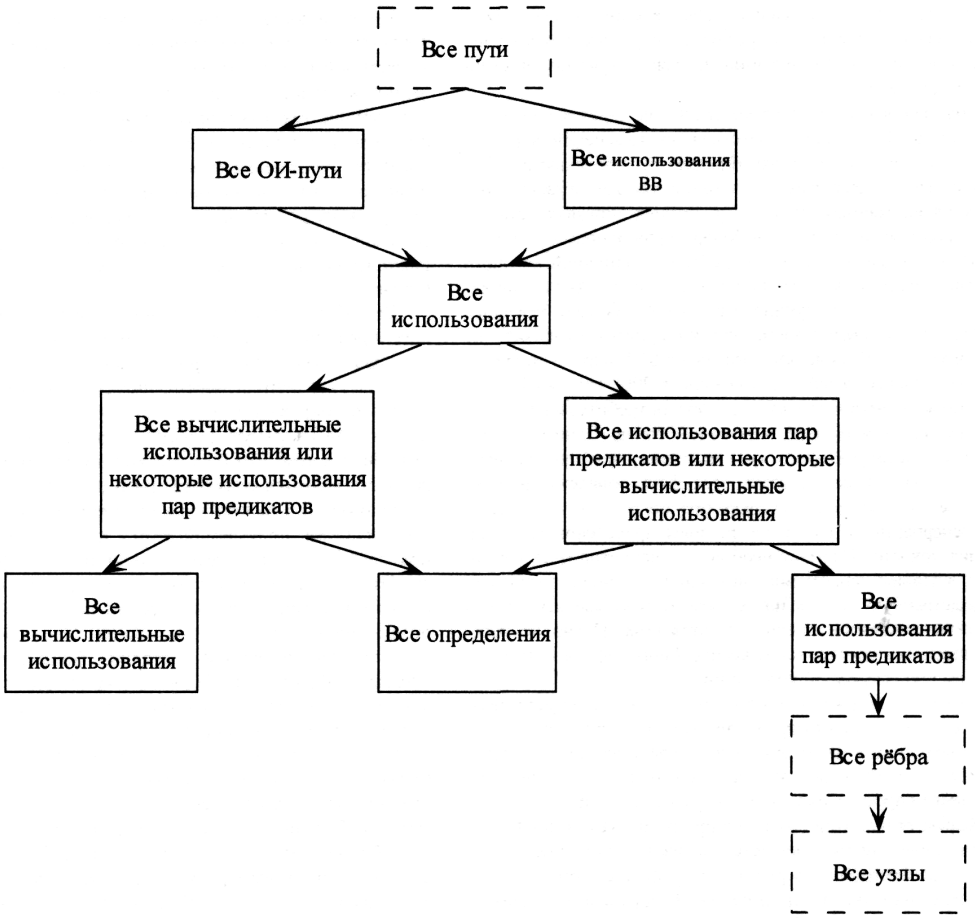


Рис. 2. *Иерархия критериев покрытия*

Если выбран более строгий критерий покрытия теста, число ассоциаций определение-использование, которые необходимо протестировать, увеличивается. Например, для удовлетворения критерия все-определения необходимо тестировать только четыре ассоциации, тогда как для удовлетворения критерия все-использования необходимо тестировать все ассоциации.

Минимальный набор ассоциаций, который необходимо тестировать для удовлетворения данного критерия, не всегда уникален. Например, для удовлетворения критериев все-определения может быть выбрана любая из первых девяти ассоциаций для тестирования определения x в узле 1. Требования к тесту или тестовые случаи могут отдавать предпочтения предыдущей характеристике. В общем, такие предпочтения могут использоваться для уменьшения усилий для тестирования либо для увеличения коэффициента обнаружения ошибок [4].

Тестирование большего количества ассоциаций не всегда требует исполнения большего количества тестовых случаев. В данном примере, несмотря на то, что минимальное количество ассоциаций определение-использование, которые должны быть протестированы для удовлетворения различных критериев, сильно различаются, все критерии покрытия удовлетворяются лишь двумя тестовыми случаями. Фактически было показано, что критерии все-использования, потому что требуется относительно небольшое количество тестовых случаев для удовлетворения этого критерия [5].

Хотя может быть возможно найти минимальное количество полных путей, удовлетворяющих данный критерий покрытия теста, данный подход не может всегда быть использован для минимизации усилий для тестирования. Это происходит из-за того, что некоторые полные пути могут быть недостижимы. Например, полный путь 1-2-4-8 для программы на рисунке 3 недостижим.

Анализ движения данных в качестве основы системы обнаружения уязвимостей удобен в плане внедрения в современные компиляторы. Большинство современных компиляторов преобразуют программу к виду единичных статических присваиваний (Static Single Assignment Forms (SSA)) до этапа оптимизации. В форме SSA каждая переменная в программе целью для присваивания лишь один раз. Такое преобразование осуществляется путём создания множества версий переменной, по одной для каждого присваивания, а также предоставления так называемых фи-функций в точках соединения множества версий. SSA форма программы позволяет производить многие оптимизации более эффективно, а также сильно упростить применение этих оптимизаций. Так как несвязанным определениям каждой переменной назначаются новые имена переменных, алгоритмам анализа движения данных не нужно исследовать их взаимосвязь. Таким образом, внедрение анализа движения данных на данном этапе является эффективным.

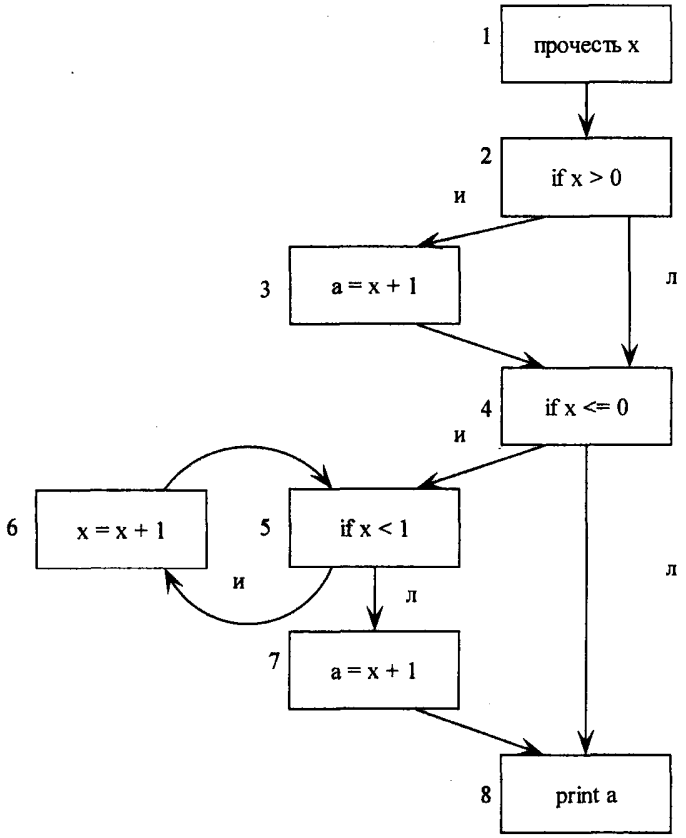
СПИСОК ЦИТИРОВАННЫХ ИСТОЧНИКОВ

1. Michael Sutton. 2007. Fuzzing. Brute force vulnerability discovery. Addison-Wesley.
2. Jack Koziol. 2004. The Shellcoder's Handbook. Wiley publishing.
3. E.Duesterwald, R.Gupta, M.L.Soffa. Rigorous Data Flow Testing through Output Influences, 2nd Irvine Software Symposium, March 1992.
4. T.L. Graves. An empirical study of regression test selection techniques, Software Eng. Methodol., 2001
5. E.J. Weyuker, The cost of data flow testing: an empirical study, IEEE Trans. Software Eng., 1990.

Материал поступил в редакцию 22.01.08

DROGIN V.V. Application at a stage of compilation of automatic testing of the software for detection of mistakes

In given clause is opened applications fuzzing at a stage of compilation. Fuzzing or automatic testing of the programs for presence of mistakes by a method of rough force. The economic conditionality of application fuzzing is mentioned, and also some trial and error methods of test cases for testing the software are resulted as an example for presence of mistakes.



Тестовый ввод	Полный путь
x = 1	P_1: 1-2-3-4-8
x = -1	P_2: 1-2-4-5-6-5-6-5-7-8

Критерии→ Ассоциация↓	Все определения	Все вычислительные использования	Все использования пар предикатов	Все вычислительные использования или некоторые использования пар предикатов	Все использования пар предикатов или некоторые вычислительные использования	Все использования
(1, (2, и), x)	u		u		u	u
(1, (2, л), x)			u		u	u
(1, 3, x)		u		u		u
(1, (4, и), x)			u		u	u
(1, (4, л), x)			u		u	u
(1, (5, и), x)			u		u	u
(1, (5, л), x)			u		u	u
(1, 6, x)		u		u		u
(1, 7, x)		u		u		u
(3, 8, a)	u	u		u	u	u
(7, 8, a)	u	u		u	u	u
(6, 6, x)	u	u		u		u
(6, 7, x)		u		u		u
(6, (5, и), x)			u		u	u
(6, (5, л), x)			u		u	u
Удовлетворяющие пути	$\{P_1,P_2\}$	$\{P_1,P_2\}$	$\{P_1,P_2\}$	$\{P_1,P_2\}$	$\{P_1,P_2\}$	$\{P_1,P_2\}$

Рис. 3. *Пример критериев тестирования*