УДК 004.052.2

Б. В. СИВКО, магистр технических наук, Белорусский государственный университет транспорта, г. Гомель

РАЗРАБОТКА БЕЗОПАСНЫХ И ОТКАЗОУСТОЙЧИВЫХ СИСТЕМ НА ОСНОВЕ ВЗАИМНОЙ ПРОВЕРКИ АКСИОМАТИЧЕСКИХ БАЗИСОВ

Предложен метод, позволяющий формализованно проводить проектирование и верификацию систем, обнаруживающих собственные отказы. Рассмотрены самопроверка и самотестирование на основе предложенного метода. Приведены примеры применения. Введено понятие полноты проверки и условие её выполнения. Показано, что полнота проверки может являться как условием полной самопроверяемости системы, так и целевым свойством при разработке.

Важной задачей в разработке и верификации безопасных и отказоустойчивых систем является обнаружение отказов с последующей реакцией, позволяющей перевести систему в безопасное состояние или запустить процесс её восстановления. В настоящее время данная проблема не имеет единого решения, поэтому актуальной является разработка методов и средств, позволяющих эффективно решать задачу обнаружения отказов [1]. В статье предлагается решение с помощью аксиоматических базисов.

Аксиоматическим базисом (далее – базис) является набор утверждений, на основе истинности которых происходит разработка или верификация. Разработка функциональности системы на различных базисах позволяет формализованно создавать отказоустойчивые и безопасные системы (подробнее – в работе [2]).

Определение функции корректности. Введем функцию корректности работы системы *s(x),* которая принимает истинное значение в случае, когда система находится в безопасном и работоспособном состоянии. Если рассмотреть работу системы в виде дискретных переходов между её состояниями, то относительно отказов и функции *s(x)* возможны три варианта поведения, показанные на рисунке 1.

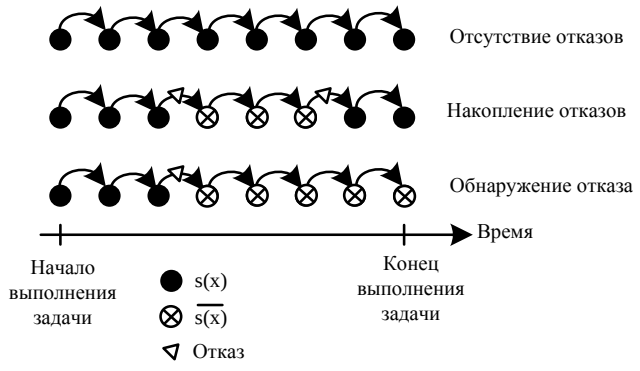

Рисунок 1 – Обнаружение отказов

Система начинает работу с некоторого корректного состояния, когда *s(x)* истинно. В случае отказа к моменту завершения выполнения задачи может получится так, что система вернется в состояние, когда выполняется *s(x),* что ведет к накоплению отказов. Если в конце выполнения *s(x)* не выполняется, то происходит обнаружение отказа.

Рассмотрим систему, которая разрабатывается или верифицирована на базисе A_f, состоящего из трех утверждений A_1, A_2 и A_3, и ситуацию, когда может произойти отказ, в результате которого нарушится базис A_3. Ключевой идеей обнаружения отказа с помощью аксиоматических базисов является то, что в результате нарушения базиса утверждение $\overline{A}_3$, станет истинным, а на этом основании можно построить функциональность так, что отказ будет обнаружен. Иллюстрация проверки базиса с обнаружением отказа показана на рисунке 2.

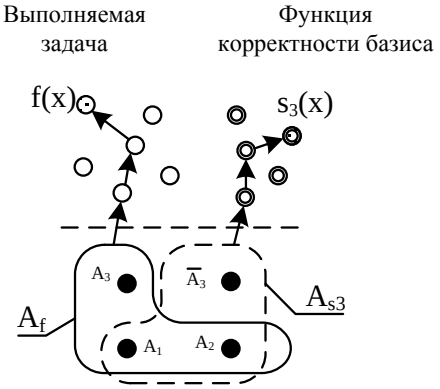

Рисунок 2 – Проверка базиса

Для системы, показанной на рисунке 2, по аналогии определяются функции $s_1(x)$, $s_2(x)$ и $s_3(x)$, которые проверяют соответствующий базис $A_i(x)$, т. е. для проверки корректности работы системы нужно рассмотреть ряд функций, каждая из которых проверяет некоторые утверждения базиса таким образом, чтобы все функции проверили весь базис, но при этом были независимы от отказов, влияющих на утверждения базиса.

Другими словами, система выполняет функцию $f(x)$ на основании базиса A_f, и при этом имеется функция безопасности $s(x)$, которая реагирует на факт отказа, нарушающего хотя бы одно из условий A_1, A_2 или A_3. В отличие от $f(x)$, *s(x)* построена на базисах A_{si}, и иммунитентна к соответствующим проверяемым отказам.

Самопроверка и самотестирование. Проверка базиса может проводиться различными способами. Рассмотрим два из них: полной проверки и проверки выполнения каждого действия.

Полная проверка базиса представляет собой проверку всех утверждений и их возможных значений аргументов для системы, которые могут потребоваться при выполнении задачи. Например, базис для операции AND (логическое «И») может быть сформулирован следующим образом:

$$A_{and} \equiv (\ AND(0,\ 0) = 0\ ) \wedge (\ AND(0,\ 1) = 0\ ) \wedge (\ AND(1,\ 0) = 0\ ) \wedge (\ AND(1,\ 1) = 1\ ). \quad (1)$$

Условие (1) можно функционально проверить до и после выполнения задачи, завершающейся за конечное время, которое достаточно мало для того, чтобы возникло накопление отказов (см. рисунок 1). Также данное условие может проверяться периодически, т. е. можно проводить самотестирование базиса.

Проверка выполнения каждого действия, в отличие от полной проверки, представляет собой параллельный расчет на базисе A_{s3}, запускаемый на тех же данных. Как следствие, не требуется проверять корректность всего базиса, а нужно тестировать только те операции и на тех данных, которые были задействованы. В этом случае базис операции AND может быть сформулирован другим образом:

$$A_{and} \equiv AND(x, y) = (x \wedge y). \qquad (2)$$

Функционально пример проверки операции AND показан на рисунке 3.

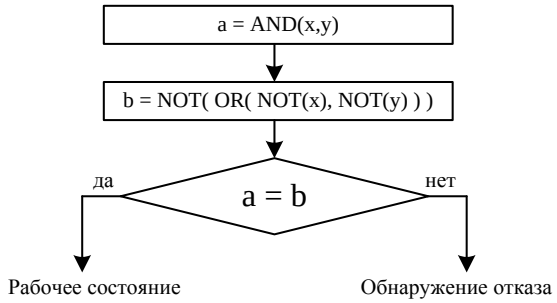

Рисунок 3 – Проверка выполнения действия

В случае такой организации контроля происходит самопроверка базисов, когда выполняется вычисление дважды на разных базисах, и в дальнейшем результаты сравниваются.

Каждый из двух описанных способов проверки имеют свои достоинства и недостатки. Полная проверка требует предельного времени выполнения задачи, и не всегда возможно осуществить полный перебор для проверки всего условия базиса. Проверка выполнения каждого действия в реальном времени не нуждается в полном переборе, но не всегда осуществима и требует более сильной синхронизации [3]. Отличительной особенностью подхода с помощью аксиоматических базисов является то, что он предоставляет формализованный инструмент для определения функции, проверяющей корректность работы системы.

Отдельным способом обнаружения отказов на основе аксиоматических базисов является использование условия, что утверждение базиса перестает быть истинным (например, когда выполняется $\overline{A}_3$, см. рисунок 2). Другими словами, когда утверждение базиса нарушается, то некоторое другое утверждение становится истинным, и это свойство можно использовать для построения отказоустойчивых и безопасных систем.

Пример. Микропроцессорная система работает по программе, которая записана в памяти, а обращение к ней выполняется посредством программного счетчика, являющегося регистром, состоящим из 8 бит. Для обеспечения отказоустойчивости требуется защититься от отказов, изменяющих биты программного счетчика в постоянные 0 и 1. В качестве решения поместим по адресам 01010101 и 10101010 специальные команды, выполнение которых говорит о том, что система работает корректно (например, это может быть передача сигнала на внешнюю систему). Во время выполнения циклического алгоритма за конечное время происходит периодический вызов команд по данным адресам. Работа алгоритма показана на рисунке 4.

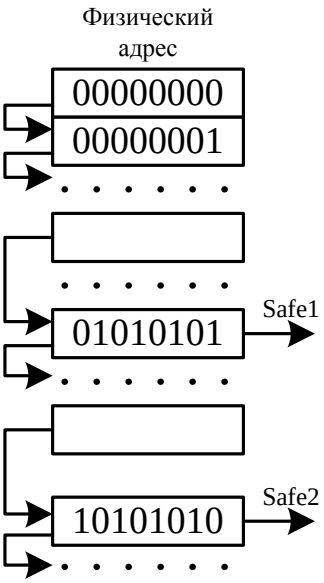

Рисунок 4 – Защита программного счетчика

Основная функциональность работает таким образом, что программный счетчик корректен. Но при этом функция, проверяющая базис, основывается на предположении, что в случае отказа один из адресов 01010101 или 10101010 будет недоступен, и, как результат, один из сигналов *Safe1* или *Safe2* будет отсутствовать, что переведет систему в безопасное состояние. Защита от отказов основывается на том, что изначально в системе утверждение о доступности адресов выполняется, но появление отказа приводит к тому, что начинает выполняться утверждение о том, что один из адресов недоступен, и за счет этого происходит обнаружение отказа.

Полнота проверки. Для проведения отказоустойчивой проверки базисов необходимо проводить её таким образом, чтобы каждый базис проверялся не зависящими от него средствами, т. е. на другом базисе. Поэтому одной из задач является приведение базисов и проверяющих их функций к такому виду, когда проверка, обладающая описанными свойствами, возможна.

Для выполнения полноты проверки требуется рассматривать как базисы, так и проверяющие их функции. Рассмотрим систему, работающую на основании n базисов A_i, а также функции $s_i(x)$, проверяющие соответствующий базис A_i, и базисы A_{si}, на основании которых реализованы соответствующие функции $s_i(x)$. Для полноты проверки необходимо, чтобы выполнялись два условия.

1 Если перестал быть истинным базис A_i, то должен выполняться проверяющий его базис A_{si}.

2 Каждая функция $s_i(x)$ должна быть реализуема на функциональности, предоставляемой базисом A_{si}.

В случае выполнения полной проверки система обладает свойством самопроверяемости в рамках рассматриваемой теории. Другими словами, если аксиоматиче-

ские допущения надежно защищены, а разработка и верификация прошли без ошибок, то система всегда обнаружит отказ.

Следует отметить, что выполнение условия полной проверки является идеальным конечным результатом, который говорит о факте полной самопроверяемости системы в рамках рассматриваемых допущений. Т. е., данное условие можно использовать в качестве целевого утверждения, подлежащего выполнению.

Рассмотрим систему, обладающую полнотой проверки, которая выполняет функцию f на основании двух базисов A_1 и A_2. Для проверки базисов есть две функции s_1 и s_2, проверяющие A_1 и A_2 соответственно. Исходя из построения, функцию s_1 необходимо реализовать на базисе A_2, а s_2 на базисе A_1. Зависимости между описанными сущностями показаны на рисунке 5.

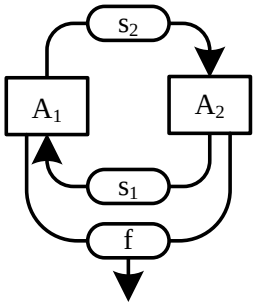

Рисунок 5 – Система со свойством полной проверки

В качестве примера такой системы можно рассмотреть микроконтроллер PIC [4], где первым базисом является корректность работы аккумулятора (W) и флага переполнения (C), а вторым – корректность ячейки памяти *X*. Кроме того, базисы имеют различный набор команд. В случае обнаружения отказа происходит переход по метке *SAFE*. Проверка происходит на отказ константной единицы. Функции s_1 и s_2 базисов показаны в таблице 1.

Если функция f может быть реализована на каждом из базисов отдельно, то в этом случае можно построить систему с более высокими показателями надежности, когда в случае отказа одного из базисов второй берет выполнение функций на себя.

Следует отметить, что увеличение количества базисов без надлежащих мероприятий по их изоляции друг от друга может вести к отказу по общей причине. Это обусловлено тем, что разработка и доказательство безопасности с помощью аксиоматических базисов опираются на утверждение о том, что отказы в различных базисах независимы. Поэтому одной из задач является выбор таких базисов, которые наименее подвержены отказам по общей причине.

Таблица 1 – **Программная проверка базисов**

Функция	Программа	
s_1 (приведена проверка одного бита из 8)	btfsc	W,0
	bsf	X,0
	btfss	X,0
	goto	SAFE
s_2	clrf	X
	movf	X, 0
	addlw	255
	movf	STATUS, 0
	andlw	1
	addlw	PCL, 1
	goto	OK
	goto	SAFE

Математически любое доказательство свойств систем по отказоустойчивости базируется на некоторых утверждениях, например на предположении о нулевой задержке, на независимости отказов, на предполагаемой интенсивности отказов и др., которые являются достаточно малым базисом, и при таких обстоятельствах доказательство свойств отказоустойчивости может быть сложной задачей. Предложенный метод, основывающийся на взаимной проверке аксиоматических базисов, основывается на том, что отказы в двух базисах происходят независимо. Это позволяет сформировать два больших множества утверждений, на которых проведение взаимной проверки и последующее доказательство свойств отказоустойчивости облегчаются.

Заключение. Таким образом, предложенный метод позволяет формализованно создавать и верифицировать системы, которые способны обнаруживать факт отказа и, как следствие, переходить в безопасное состояние или самовосстанавливаться, что позволяет выйти на новый уровень формализации и качества в вопросах разработки и верификации отказоустойчивых и безопасных систем.

Список литературы

1 **Бочков, К. А.** Микропроцессорные системы автоматики на железнодорожном транспорте : учеб. пособие / К. А. Бочков, А. Н. Коврига, С. Н. Харлап // Гомель, БелГУТ. – 2013.

2 **Сивко, Б. В.** Диверситетные аксиоматические базисы для разработки безопасных и отказоустойчивых систем / Б. В. Сивко // Вестник БелГУТа: Наука и Транспорт. – 2014. – № 1(28). – С. 19–23.

3 **Brilliant, S.** The Consistent Comparison Problem in N-Version Programming / S. Brilliant, J. C. Knight, N. G. Leveson // IEEE Trans. on Software Engineering. – 1989. – Vol. SE-15, No. 11.

4 **Verle, M.** PIC Microcontrollers / M. Verle // Mikro-Elektronika, 1st edition. – 2008. – 394 p.

Получено 27.03.2015

B. V. Sivko. The axiomatic basis mutual checking method for development of safety and fault-tolerant systems.

The method of self-fault detection is introduced. The method can be used to design and verify fault-tolerance systems. Self-testing and self-checking are considered on the method basis. Examples of using are shown. Introduced the concept of ‘completeness of checking’ and its formal condition. It is shown, the completeness of checking can be as a full self-checking condition as a target property during development.