Робастное, адаптивное и сетевое управление

**© 2021 г. А.М. РОМАНКЕВИЧ, д-р техн. наук (romankev@scs.kpi.ua),
К.В. МОРОЗОВ (mcng@ukr.net),
В.А. РОМАНКЕВИЧ, д-р техн. наук (romankev@scs.kpi.ua)
(Национальный технический университет Украины
«Киевский политехнический институт им. Игоря Сикорского»)**

ФОРМАЛЬНЫЙ МЕТОД ОПРЕДЕЛЕНИЯ СОСТОЯНИЯ ПРОЦЕССОРОВ МНОГОПРОЦЕССОРНОЙ СИСТЕМЫ ПРИ ЕЕ ТЕСТИРОВАНИИ

Работа посвящена проблеме самотестирования многопроцессорных систем в рамках модели Препараты–Метца–Чена. Предлагается метод формализации процедуры установления состояния всех процессоров M-диагностируемой системы на основе анализа результатов некоторого множества взаимных тестовых проверок процессорами. Анализ сводится к решению булева уравнения, составленного на основе результатов этих проверок, и позволяет установить либо состояние (исправен/неисправен) всех процессоров системы, либо тот факт, что количество выполненных проверок для этого недостаточно (т.е. состояние каких-то конкретных процессоров не определено). Сказанное остается справедливым в тех случаях, когда число неисправных процессоров в системе не превышает величины M.

Ключевые слова: многопроцессорные системы, диагностический граф, взаимное тестирование процессоров, ПМЧ-модель.

DOI: 10.31857/S0005231021030065

1. Введение и постановка задачи

Многопроцессорные системы (МС) находят все более широкое применение в современной технике и особенно в системах управления сложными объектами. Важной частью работы МС является процесс ее тестирования с целью определения состояния (исправен/отказал) каждого из процессоров. Эта информация может позволить гибко и эффективно использовать имеющиеся резервы и реконфигурировать систему для обеспечения продолжения ее функционирования [1, 2].

Состояние каждого из процессоров может определяться в результате выполнения им некоторого набора тестов и сравнения полученных результатов с эталоном. Если для некоторого процессора результаты сравнения не совпадают, то он определяется как неисправный. Тестирование процессоров может производиться как отдельным специализированным узлом (например, в процессе изготовления), так и в результате их взаимного тестирования (в процессе эксплуатации), когда тестирующий процессор определяет состояние тестируемого. В соответствии с моделью, предложенной в [3], называемой моделью Препараты–Метца–Чена (ПМЧ), результатом тестирования процессора x исправным процессором y будет 1, если в процессе тестирования были

обнаружены ошибки, или 0, если ошибок обнаружено не было. Если же тестирующий процессор y в момент тестирования был неисправен, то в результате может быть получено как 0, так и 1 вне зависимости от состояния тестируемого процессора x. Как и во множестве работ, здесь предполагается, что каналы связи между процессорами исправны, так как вероятность выхода их из строя относительно низкая.

Топология реальных систем не всегда позволяет организовать тестирование каждого процессора любым другим процессором. Эти ограничения могут быть вызваны, в частности, отсутствием непосредственных связей между процессорами, ограничением объема памяти для хранения тестовых программ и эталонных значений результатов тестов (если в системе находятся процессоры разных типов) и др. Возможность тестирования процессоров друг другом можно представить в виде ориентированного графа G, каждая из вершин которого соответствует некоторому процессору, а наличие дуги из некоторой вершины a в некоторую вершину b означает возможность тестирования процессора, соответствующего вершине b, процессором, соответствующим вершине a. Такой граф фактически отражает возможности системы в плане диагностирования.

Для любой МС имеет место t-диагностируемость [4]. Система является t-диагностируемой, если, зная результаты взаимного тестирования процессоров, можно установить состояние всех ее процессоров в случае выхода из строя не более t любых из них.

На выполнение самотестирования система затрачивает определенные ресурсы, поэтому желательно организовать этот процесс так, чтобы проводилось минимальное количество тестовых испытаний. Это позволяет уменьшить общее время тестирования, а чем чаще будет повторяться процесс тестирования, тем быстрее новый отказ будет обнаружен и тем быстрее система может быть реконфигурирована. Иными словами, минимизация количества проводимых тестов является важной задачей [5, 6] что справедливо и для других задач тестирования [7–9].

Будем считать, что процессор может в один и тот же момент либо тестировать другой процессор, либо тестироваться другим процессором. В [10] был предложен алгоритм, позволяющий определить состояние системы путем выполнения не более чем $N + 2p$ тестовых проверок, где N — количество процессоров в системе, а p — число реально неисправных процессоров. Необходимое условие применения этого алгоритма: орграф, отражающий возможности взаимотестирования процессоров МС, должен быть полным. Реальные же структуры далеко не всегда соответствуют полному графу.

В [3] определена теоретическая граница величины t и доказано, что полное диагностирование системы, т.е. определение исправности всех процессоров системы, всегда возможно, если допустимое число неисправных процессоров не превышает величины $[(N-1)/2]$, где N — количество процессоров системы. В [11] показано, что практически всегда это возможно и при большем числе неисправностей.

Во многих случаях определение состояния процессоров системы на основе результатов выполненных взаимных тестовых проверок является довольно сложной и трудоемкой процедурой.

В данной работе решается задача формализации процедуры установления состояния всех процессоров после выполнения определенного количества тестовых испытаний (не обязательно всех возможных) при любой топологии связей (не обязательно полный граф) системы.

Следует отметить, что задача построения полного теста в данной работе не рассматривается, т.е. предполагается, что некоторый набор тестовых проверок задан разработчиком системы.

2. Формальное представление результатов отдельных тестовых проверок

Остановимся на поставленной задаче подробнее. Пусть M-диагностируемая система состоит из N процессоров, и состоянию каждого из них соответствует некоторая булева переменная x_i $(i = 1, \ldots, N)$, принимающая значение 1, если процессор исправен, и 0, если он вышел из строя. В некоторый момент времени часть процессоров исправна, и соответствующие им переменные равны 1. Множество этих переменных обозначим как W. Остальные процессоры неисправны, соответствующие им переменные равны 0, и множество этих переменных обозначим через F. Очевидно, что

$$\left(\bigwedge_{x_i \in W} x_i\right) \wedge \left(\bigwedge_{x_j \in F} \overline{x}_j\right) = 1. \tag{1}$$

Отметим здесь, что если убрать, добавить, либо переставить хотя бы одну инверсию в левой части уравнения (1), оно примет нулевое значение.

Построим таблицу значений результатов тестирования j-го процессора i-м процессором в зависимости от состояний этих процессоров в соответствии с моделью ПМЧ (см. табл. 1).

Таблица 1

x_i	x_j	Результат тестирования
1	1	0
1	0	1
0	1	*
0	0	*

Символом «*» здесь обозначено произвольное значение: 0 либо 1, определяемое моделью ПМЧ.

Запишем основные уравнения, которые лягут в основу решения поставленной задачи.

В соответствии с уравнением (1) и табл. 1 результат тестирования *может быть равен* 0 лишь в двух случаях: оба процессора исправны либо тестирующий процессор неисправен, т.е. когда справедливо

$$x_i x_j \vee \overline{x}_i \equiv x_j \vee \overline{x}_i = 1. \tag{2}$$

В то же время результат тестирования *может быть равен* 1 лишь в двух случаях: если тестирующий процессор исправен, а тестируемый — неисправен либо если тестирующий процессор неисправен, т.е. когда справедливо

$$x_i\overline{x}_j \vee \overline{x}_i \equiv \overline{x}_j \vee \overline{x}_i = 1. \tag{3}$$

3. Выполнение некоторого множества тестовых проверок

Перейдем к собственно формализации. Пусть после проведения $k-1$ тестовых проверок построено некое уравнение вида $V_{k-1}(x_1,\ldots,x_N)=1$. И пусть k-й тест заключается в тестировании j-го процессора i-м процессором. В результате этого теста получено значение r_k. Можем записать

$$R_k(x_i,x_j) \triangleq \begin{cases} x_j \vee \overline{x}_i, \text{ если } r_k = 0 \\ \overline{x}_j \vee \overline{x}_i, \text{ если } r_k = 1, \end{cases} \tag{4}$$

а также

$$V_k(x_1,\ldots,x_N) \triangleq V_{k-1}(x_1,\ldots,x_N) \wedge R_k(x_i,x_j). \tag{5}$$

В соответствии с (2)–(4) видим, что $R_k(x_i,x_j)=1$, следовательно,

$$V_k(x_1,\ldots,x_N) = 1. \tag{6}$$

Далее, используя результаты следующего тестового испытания, получим выражение R_{k+1} и соответственно V_{k+1}. Продолжаем этот процесс до тех пор, пока не будут учтены результаты всех выполненных тестовых проверок.

В качестве начального выражения будем использовать $V_0(x_1,\ldots,x_N) \triangleq 1$, и, следовательно, $V_1 \triangleq R_1$. После проведения n тестов (где, вообще говоря, должны будут участвовать хотя бы по разу все N процессоров) в результате итеративного применения формулы (5) выражение V_n примет вид

$$V_n \triangleq R_1 \wedge R_2 \wedge \ldots \wedge R_n. \tag{7}$$

Покажем, что решение этого уравнения дает ответ на поставленную в работе задачу.

Действительно, преобразовав выражение (7) в совершенную дизъюнктивную нормальную форму (СДНФ), получим:

$$V_n \equiv C_1 \vee C_2 \vee \ldots \vee C_K. \tag{8}$$

Таким образом, имеем уравнение:

$$C_1 \vee C_2 \vee \ldots \vee C_K = 1. \tag{9}$$

Каждое из выражений C_i является конституентой единицы — элементарной конъюнкцией, которая содержит все переменные x_i $(i=1,\ldots,N)$. Очевидно, что только одна из конституент равна 1, и в соответствии с (1) она идентифицирует состояние системы (т.е. представляет состояние каждого из ее процессоров). Докажем, что она будет присутствовать в (9).

Вспомним, что начальный (вырожденный) вариант уравнения (до выполнения любых проверок) имеет вид $1=1$, где левая (или правая) часть может

быть выражена как дизъюнкция всех возможных конституент единицы, среди которых безусловно будет и отвечающая актуальному состоянию системы. Далее, в соответствии с результатами тестов формируются выражения R_i, позволяющие исключить некоторые из конституент (причем часто группами), приближая процесс к единственному решению.

Рассмотрим выражение V_n, полученное после проведения n тестов, представленное в СДНФ (9). Допустим, что данное выражение действительно содержит конституенту, соответствующую реальному состоянию системы, которую обозначим как C_R. Пусть на основе результатов $(n+1)$-го теста было сформировано выражение R_{n+1}. Тогда в соответствии с (5) и (9)

$$V_{n+1} \triangleq V_n \wedge R_{n+1} \equiv (C_1 \vee C_2 \vee \ldots \vee C_K) \wedge R_{n+1} \equiv \equiv C_1 \wedge R_{n+1} \vee C_2 \wedge R_{n+1} \vee \ldots \vee C_K \wedge R_{n+1}. \tag{10}$$

Очевидно, что для конституент C_i возможны только два варианта:

1) $C_i \wedge R_{n+1} \equiv C_i$, что соответствует ситуации, когда R_{n+1} не исключает состояния системы, соответствующего C_i.

2) $C_i \wedge R_{n+1} \equiv 0$, что соответствует ситуации, когда R_{n+1} такое состояние исключает.

Покажем, что конституента C_R, отвечающая реальному состоянию системы, исключена быть не может. Отметим, что в соответствии с (1) справедливо

$$C_R = 1. \tag{11}$$

Предположим, что выражение R_{n+1} исключает состояние, соответствующее C_R, т.е.

$$C_R \wedge R_{n+1} \equiv 0. \tag{12}$$

Вспомним, что в соответствии с (2)–(4) справедливо

$$R_{n+1} = 1. \tag{13}$$

Объединив (11) и (13), получим $C_R \wedge R_{n+1} = 1$. В совокупности с (12) это приводит к противоречию вида $0 = 1$, т.е. высказанное предположение неверно. Обобщая вывод для любого V_i , можно утверждать, что ни одно тестовое испытание не приведет к исключению конституенты C_R, отвечающей реальному состоянию системы, и, следовательно, $C_R \wedge R_{n+1} \equiv C_R$, и выражение V_{n+1} в СДНФ также будет содержать конституенту C_R.

Далее обратим внимание на следующее: система M-диагностируема, а потому можно определить ее состояние, если в ней вышло из строя не более чем M процессоров. Предположим, что это так, т.е. в ней действительно вышло из строя M или меньше процессоров. Тогда в соответствии с (1) любая C_j, содержащая более чем M инверсий, будет равна нулю. Иными словами, можно исключать из V_n все C_j, содержащие больше чем M инверсий. Отметим, что конституента C_R, отвечающая реальному состоянию системы, на данном этапе также не будет исключена (если только она не содержит более чем M инверсий, что соответствует ситуации, когда в системе вышло из строя больше чем M процессоров).

В результате после такого преобразования уравнение (9) примет одну из трех форм:

1) $C_i = 1$;

2) $C_{i_1} \vee C_{i_2} \vee \ldots \vee C_{i_L} = 1$;

3) $0 = 1$.

В первом случае в левой части уравнения получена ровно одна конституента единицы, т.е. элементарная конъюнкция, содержащая все переменные x_i $(i = 1, \ldots, N)$. В этом случае, если в системе действительно вышло из строя не более чем M процессоров (т.е. если конъюнкции с большим количеством инверсий были исключены правомерно), в соответствии с (1) возможно определить состояние каждого из процессоров, т.е. поставленную задачу можно считать решенной. Действительно, в соответствии с методом построения и решения булева уравнения можно сказать: исправны все те процессоры, которым соответствуют переменные, входящие в конъюнкцию без инверсий, и неисправны все те процессоры, которым соответствуют переменные, входящие в конъюнкцию с инверсиями.

Во втором случае в выражении осталось более одной элементарной конъюнкции. Если в системе действительно вышло из строя не более чем M процессоров, то можно лишь сказать, что одна из этих конъюнкций действительно соответствует реальному состоянию системы (как уже было показано, она не может быть исключена в соответствии с результатами каких-либо тестов). Иными словами, имеет место некая неопределенность, т.е. тест, выбранный разработчиком, недостаточен для полного диагноза состояний процессоров.

Здесь следует отметить, что для установления состояния системы разработчик может провести дополнительные тестовые проверки, результаты которых могут исключить остальные конъюнкции и тем самым позволят разрешить данную неопределенность. Это могут быть проверки, осуществляемые исправными (они во всех конъюнкциях присутствуют без инверсий) процессорами (см. пример ниже), поскольку таким результатам можно верить в соответствии с ПМЧ-моделью, или какие-то другие, после чего разработчик может повторить предлагаемый анализ. Если же уже были выполнены все тесты, возможные в соответствии с топологией системы, то данный результат, очевидно, означает, что система не является M-диагностируемой.

Третий случай, когда из выражения исключаются все элементарные конъюнкции, может наступить только тогда, когда в системе на самом деле вышло из строя более чем M процессоров (следовательно, конъюнкция, соответствующая реальному состоянию системы, была исключена неправомерно, что, в свою очередь, привело к отсутствию конъюнкций в левой части уравнения). Отметим, что обратное, вообще говоря, не верно: в случае выхода из строя более чем M процессоров уравнение может принимать любую из представленных выше форм.

Таким образом, можем сформулировать следующий алгоритм определения состояния M-диагностируемой системы:

1. На основе результатов очередных l тестовых испытаний сформировать выражения R_{k+1}, R_{k+2}, $\ldots$, R_{k+l} в соответствии с (4).

2. Дополнить выражение V_k, полученное на предыдущих итерациях, до выражения V_{k+l} в соответствии с (5). Для первой итерации в качестве V_0 использовать «1».

3. Преобразовать полученное выражение в СДНФ, исключая все конъюнкции, содержащие более чем M инверсий.

4. Если полученное выражение содержит больше одной конъюнкции, то провести одну или несколько дополнительных тестовых проверок и перейти к п. 1. Если дополнительные тесты провести невозможно, т.е. не проведенных допустимых тестовых проверок не осталось, то система — не M-диагностируема.

5. Если полученное выражение содержит ровно одну конъюнкцию, то определить в соответствии с ней состояние системы, считая процессоры, соответствующие переменным, входящим в нее без инверсий, исправными, а остальные — неисправными.

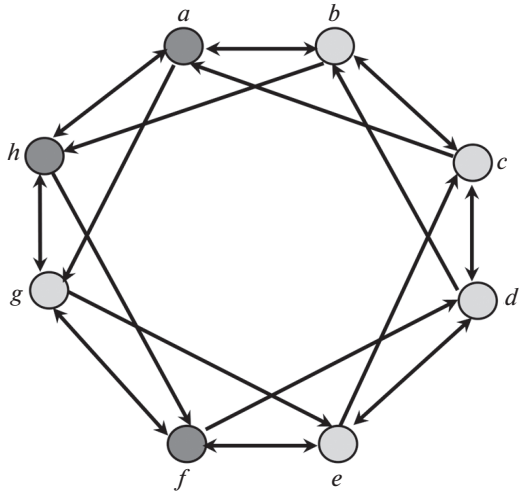
6. Если полученное выражение не содержит конъюнкций, т.е. если в результате проведенных преобразований был получен «0», то в системе вышло из строя больше чем M процессоров.

Рассмотрим пример, иллюстрирующий описанный метод. Используется модель ПМЧ, и число отказов не превышает половины числа процессоров.

4. Пример

Рассмотрим случай: 3-диагностируемая система состоит из 8 процессоров (обозначим их как a, b, c, d, e, f, g, h). Граф, представляющий топологию связей, представлен на рисунке. Пусть вышли из строя 3 процессора: a, f и h.

Пусть проведены следующие тестовые проверки: $a \to b$, $c \to d$, $e \to f$, $g \to h$, $b \to c$, $d \to e$, $f \to g$, $h \to a$, $c \to a$, $d \to b$, $g \to e$, $h \to f$ (тестирующий процессор указан слева от стрелки, а тестируемый — справа) и пусть получены результаты, которые приведены в табл. 2.



Граф топологии системы.

Таблица 2

№ теста, i	Тестирующий процессор	Тестируемый процессор	Результат теста, r_i	Выражение R_i
1	a	b	0	$R_1 \triangleq b \vee \overline{a}$
2	c	d	0	$R_2 \triangleq d \vee \overline{c}$
3	e	f	1	$R_3 \triangleq \overline{f} \vee \overline{e}$
4	g	h	1	$R_4 \triangleq \overline{h} \vee \overline{g}$
5	b	c	0	$R_5 \triangleq c \vee \overline{b}$
6	d	e	0	$R_6 \triangleq e \vee \overline{d}$
7	f	g	1	$R_7 \triangleq \overline{g} \vee \overline{f}$
8	h	a	0	$R_8 \triangleq a \vee \overline{h}$
9	c	a	1	$R_9 \triangleq \overline{a} \vee \overline{c}$
10	d	b	0	$R_{10} \triangleq b \vee \overline{d}$
11	g	e	0	$R_{11} \triangleq e \vee \overline{g}$
12	h	f	0	$R_{12} \triangleq f \vee \overline{h}$

Можем записать: $V_{12} \triangleq R_1 \wedge R_2 \wedge \ldots \wedge R_{12} = 1$. Выразим V_{12} в форме СДНФ, исключая при этом все конъюнкции, содержащие более чем 3 инверсии:

$$V_{12} \triangleq R_1 \wedge R_2 \wedge \ldots \wedge R_{12} \triangleq$$

$$\triangleq (b \vee \overline{a})(d \vee \overline{c})(\overline{f} \vee \overline{e})(\overline{h} \vee \overline{g})(c \vee \overline{b})(e \vee \overline{d})(\overline{g} \vee \overline{f})(a \vee \overline{h}) \wedge$$

$$\wedge (\overline{a} \vee \overline{c})(b \vee \overline{d})(e \vee \overline{g})(f \vee \overline{h}) = ((b \vee \overline{a})(a \vee \overline{h})(\overline{a} \vee \overline{c}))((d \vee \overline{c})(c \vee \overline{b})(b \vee \overline{d})) \wedge$$

$$\wedge ((\overline{f} \vee \overline{e})(e \vee \overline{d})(e \vee \overline{g}))((\overline{h} \vee \overline{g})(\overline{g} \vee \overline{f})(f \vee \overline{h})) = (ab\overline{c} \vee \overline{a}b\overline{h} \vee b\overline{c}\overline{h} \vee \overline{a}\overline{h} \vee \overline{a}\overline{c}\overline{h}) \wedge$$

$$\wedge (bcd \vee \overline{b}\overline{c}\overline{d})(e\overline{f} \vee e\overline{f}\overline{g} \vee \overline{d}e\overline{f} \vee \overline{d}f\overline{g} \vee \overline{d}\overline{e}\overline{g})(\overline{g}\overline{h} \vee f\overline{g}\overline{h} \vee \overline{f}\overline{h} \vee f\overline{g} \vee \overline{f}\overline{g}\overline{h}) =$$

$$= (ab\overline{c} \vee \overline{a}\overline{h})(bcd \vee \overline{b}\overline{c}\overline{d})(e\overline{f} \vee \overline{d}\overline{e}\overline{g})(\overline{f}\overline{h} \vee f\overline{g}) =$$

$$= (\overline{a}bcd\overline{h} \vee \cancel{\overline{a}\overline{b}\overline{c}\overline{d}\overline{h}})(e\overline{f}\overline{h} \vee \overline{d}\overline{e}f\overline{g} \vee \cancel{\overline{d}\overline{e}\overline{g}\overline{h}}) =$$

$$= \overline{a}bcde\overline{f}g\overline{h} = 1.$$

В результате получена ровно одна конституента единицы, которая соответствует следующему состоянию системы: процессоры b, c, d, e, g — исправны, а процессоры a, f, h — неисправны. Найденное решение соответствует действительному состоянию системы.

Отметим, что, как это и было сделано в примере, исключение элементарных конъюнкций, содержащих более чем M инверсий, правомерно выполнять на любом этапе преобразования. Действительно, в результате раскрытия скобок элементарная конъюнкция порождает выражения с не меньшим, чем содержалось в ней, числом инверсий. Данный прием зачастую позволяет несколько упростить процесс преобразования.

Стоит также отметить, что рассмотренный для данного примера набор тестовых проверок является достаточным для получения однозначного ответа лишь для представленной комбинации результатов тестов. Так, в случае, если бы результаты 8-й и 12-й проверок отличались от представленных (т.е. $r_8 = 1$ и $r_{12} = 1$, что также соответствует модели ПМЧ для рассматриваемого состояния системы), то имело бы место $R_8 \triangleq \overline{a} \vee \overline{h}$ и $R_{12} \triangleq \overline{f} \vee \overline{h}$. Тогда преобразование V_{12} было бы следующим:

$$V_{12} \triangleq R_1 \wedge R_2 \wedge \ldots \wedge R_{12} \triangleq$$
$$\triangleq (b \vee \overline{a})(d \vee \overline{c})(\overline{f} \vee \overline{e})(\overline{h} \vee \overline{g})(c \vee \overline{b})(e \vee \overline{d})(\overline{g} \vee \overline{f})(\overline{a} \vee \overline{h}) \wedge$$
$$\wedge (\overline{a} \vee \overline{c})(b \vee \overline{d})(e \vee \overline{g})(\overline{f} \vee \overline{h}) = ((b \vee \overline{a})(\overline{a} \vee \overline{h})(\overline{a} \vee \overline{c}))((d \vee \overline{c})(c \vee \overline{b})(b \vee \overline{d})) \wedge$$
$$\wedge ((\overline{f} \vee \overline{e})(e \vee \overline{d})(e \vee \overline{g}))((\overline{h} \vee \overline{g})(\overline{g} \vee \overline{f})(\overline{f} \vee \overline{h})) =$$
$$= (\overline{a}b \vee \overline{a}b\overline{c} \vee \overline{a}b\overline{h} \vee b\overline{c}\overline{h} \vee \overline{a} \vee \overline{a}c \vee$$
$$\vee \overline{a}\overline{h} \vee \overline{a}c\overline{h})(bcd \vee \overline{b}\overline{c}\overline{d})(e\overline{f} \vee e\overline{f}\,\overline{g} \vee \overline{d}e\overline{f} \vee \overline{d}f\overline{g} \vee \overline{d}e\overline{g})(\overline{f}\,\overline{g}\overline{h} \vee \overline{g}\overline{h} \vee \overline{f}h \vee \overline{f}\,\overline{g}) =$$
$$= (b\overline{c}\overline{h} \vee \overline{a})(bcd \vee \overline{b}\overline{c}\overline{d})(e\overline{f} \vee \overline{d}e\overline{g})(\overline{g}\overline{h} \vee \overline{f}\overline{h} \vee \overline{f}\,\overline{g}) =$$
$$= (\overline{a}bcd \vee \cancel{\overline{a}\overline{b}\overline{c}\overline{d}})(e\overline{f}h \vee e\overline{f}\,\overline{g} \vee \cancel{\overline{d}e\overline{g}h} \vee \cancel{\overline{d}e\overline{f}\,\overline{g}}) =$$
$$= \overline{a}bcde\overline{f}h \vee \overline{a}bcde\overline{f}\,\overline{g} = \overline{a}bcde\overline{f}g\overline{h} \vee \overline{a}bcde\overline{f}\,\overline{g}h \vee \cancel{\overline{a}bcde\overline{f}\,\overline{g}\overline{h}} =$$
$$= \overline{a}bcde\overline{f}g\overline{h} \vee \overline{a}bcde\overline{f}\,\overline{g}h = 1.$$

Как видим, в данном случае получено две конституенты единицы, одна из которых ($\overline{a}bcde\overline{f}g\overline{h}$) соответствует действительному состоянию системы. Это означает лишь одно: необходимы дополнительные тестовые проверки. Впрочем, даже полученный частичный результат позволяет сделать некоторые выводы. Так, в обеих конституентах переменные a и f записаны с инверсиями, а b, c, d и e — без инверсий, из чего однозначно следует, что соответствующие первым переменным процессоры неисправны, а вторым — исправны. Состояние же процессоров g и h не определено.

Рассматриваемый пример позволяет дать рекомендации общего характера: для разрешения неопределенности достаточно выполнить проверки подозреваемых процессоров каким-либо из исправных. Для рассматриваемой в примере топологии связей системы достаточно проверки $b \to h$. Действительно, будет получен результат $r_{13} = 1$ и выражение $R_{13} \triangleq \overline{h} \vee \overline{b}$. Тогда в соответствии с (5)

$$V_{13} \triangleq V_{12} \wedge R_{13} = (\overline{a}bcde\overline{f}g\overline{h} \vee \overline{a}bcde\overline{f}\,\overline{g}h)(\overline{h} \vee \overline{b}) = \overline{a}bcde\overline{f}g\overline{h} = 1.$$

Таким образом получено однозначное решение, соответствующее следующему состоянию системы: процессоры b, c, d, e, g — исправны, а процессоры a, f, h — неисправны. Как и в предыдущем случае, решение соответствует реальному состоянию системы.

5. Заключение

В статье предложен формальный метод, позволяющий установить состояния процессоров M-диагностируемой многопроцессорной системы, сводящий-

ся к решению булева уравнения, сформированного на основе результатов какого-то множества тестовых проверок. Переменные этого уравнения соответствуют состояниям каждого из процессоров. Метод предполагает формальное преобразование булева выражения с исключением всех конъюнкций, содержащих более M инверсий. В результате должно быть получено выражение, содержащее ровно одну конъюнкцию, переменные которой фактически определяют состояния процессоров. Если это не достигнуто, необходимо выполнить дополнительные тестовые проверки. Исходное множество испытаний предполагает участие всех процессоров. Справедливость сказанного выше имеет место лишь для ситуации, когда в системе действительно вышло из строя не более чем M процессоров.

Следует также отметить ряд достоинств предлагаемого метода. В частности, на этапе изготовления он позволяет выявить ситуации, когда система на самом деле не является M диагностируемой (правда, для этого в худшем случае необходимо перебрать все возможные комбинации состояний системы и результатов тестов, что может иметь неприемлемо большую сложность в общем случае, однако может оказаться эффективным для анализа систем, имеющих регулярную структуру). Кроме того, на этапе эксплуатации предложенный метод в определенных случаях позволяет выявить также факт выхода из строя более чем M процессоров.

СПИСОК ЛИТЕРАТУРЫ

1. *Ведешенков В.А.* Организация диагностирования цифровых систем со структурой симметричного двудольного графа // Пробл. управления. 2009. № 6. С. 59–67.
2. *Каравай М.Ф., Подлазов В.С.* Расширенный обобщенный гиперкуб как отказоустойчивая системная сеть для многопроцессорных систем // УБС. 2013. № 45. С. 344–371.
3. *Preparata F.P., Metze G., Chien R.T.* On the Connection Assignment Problem of Diagnosable Systems // IEEE Trans. Electron. Comput. 1967. ES-16. No. 6. P. 848–854.
4. *Hakimi S.L., Amin A.T.* Characterization of Connection Assignment of Diagnosable Systems // IEEE Trans. Comput. 1974. C-23. No. 1. P. 86–88.
5. *Романкевич В.А.* Самотестирование многопроцессорных систем с регулярными диагностическими связями // АиТ. 2017. № 2. С. 115–127.

   Romankevich V.A. Self-testing of Multiprocessor Systems with Regular Diagnostic Connections // Autom. Remote Control. 2017. V. 78. No. 2. P. 289–299.
6. *Романкевич В.А., Романкевич А.В, Ахмедова Д.Н.* Метод уменьшения количества взаимопроверок при самотестировании многопроцессорных систем // Радиоэлектронные и компьютерные системы. 2018. № 4. С. 61–66.
7. *Drozd J., Drozd A., Al-dhabi M.* A resource approach to on-line testing of computing circuits // Proc. IEEE East-West Design & Test Symposium. Batumi, Georgia, 2015. P. 276–281. https://doi.org/10.1109/EWDTS.2015.7493122
8. *Drozd A., Drozd J., Antoshchuk S., Nikul V., Al-dhabi M.* Objects and Methods of On-Line Testing: Main Requirements and Perspectives of Development // Proc. IEEE East-West Design & Test Symposium. Yerevan, Armenia, 2016. P. 72–76. https://doi.org/10.1109/EWDTS.2016.7807750

9. *Димитриев Ю.К.* О t-диагностируемости мультипроцессорных систем с симметричной циркулянтной структурой // АиТ. 2013. № 1. С. 135–145.

Dimitriev Y.K. On t-diagnosability of Multicore Systems with Symmetric Circulant Structure // Autom. Remote Control. 2013. V. 74. No. 1. P. 105–112.

10. *Белявский В.Е., Валуйский В.Н., Романкевич А.М. и др.* Самодиагностируемые многомодульные системы: некоторые оценки тестирования // АиТ. 1999. № 8. С. 148–153.

Belyavskii V.E., Valuiskii V.N., Romankevich A.M. and Romankevich V.A. Self-Diagnosable Multimodular Systems: Some Estimates of Testing // Autom. Remote Control. 1999. V. 60. No. 8. P. 1179–1183.

11. *Романкевич А.М., Романкевич В.А.* О диагностировании многопроцессорных систем при отказе более половины процессоров // АиТ. 2017. № 9. С. 84–90.

Romankevich A.M., Romankevich V.A. Diagnosis of Multiprocessor Systems under Failure of More Than Half Processors // Autom. Remote Control. 2017. V. 78. No. 9. P. 1614–1618.

Статья представлена к публикации членом редколлегии М.Ф. Караваем.

Поступила в редакцию 16.03.2020

После доработки 01.09.2020

Принята к публикации 28.10.2020