УДК 681.3.06

Бузовский О.В., д.т.н.; **Чебаненко Т.М.,** к.т.н.

СИНТЕЗ ТЕСТОВ ДЛЯ ПРОГРАММНО-УПРАВЛЯЕМЫХ ОБЪЕКТОВ ДИАГНОСТИРОВАНИЯ

Buzovskyy O.V., Chebanenko T.M. Synthesis of tests for the program-driven diagnostic objects. The model of specific class of devices offers in the article – the programmatic guided hardware components of the computer systems, that from the point of view of test control and диагностики possess in a number of properties, substantially different from the objects traditionally examined within the framework of this discipline. Formal presentation of model of the program-driven objects and model of class of disrepairs on the basis of that automation of procedures of construction of effective control and diagnostic tests is possible are examined. Complication of such procedures is estimated and quality of the tests got on the basis of these models is forecast. The paper formulated an approach to the construction of models and image transformations, which differs from the known fact that the latter takes into account the semantics. This allows formalized mathematical schemes for image analysis tasks based on their semantics. The formalization of the mathematical schemes for image analysis tasks creates the preconditions for the creation of a unified software framework, the use of which significantly reduces the cost of developing specialized software related to image analysis.

***Keywords*:** synthesis test, diagnostics, fault-class model, program-driven object

Бузовський О.В., Чебаненко Т.М. **Синтез тестів для программно-керованих об'єктів діагностування.** У статті запропоновано метод синтезу тестів для специфічного класу пристроїв – програмно керованих апаратних компонент комп'ютерних систем, які з точки зору тестового контролю і діагностики мають безліч властивостей, що істотно відрізняються від об'єктів, традиційно розглядаються в рамках цієї дисципліни. Розглядається формальне подання моделі програмно-керованих об'єктів і модель класу несправностей, на основі яких можлива автоматизація процедур побудови ефективних контрольних і діагностичних тестів. Оцінюється складність таких процедур і прогнозується якість одержуваних на основі цих моделей тестів.

***Ключові слова*:** синтез тестів, діагностика, модель класу несправностей, програмно керований об'єкт

Бузовский О.В., Чебаненко Т.М. **Синтез тестов для программно-управляемых объектов диагностирования.** В статье предложен метод синтеза тестов для специфического класса устройств – программно управляемых аппаратных компонент компьютерных систем, которые с точки зрения тестового контроля и диагностики обладают множеством свойств, существенно отличающихся от объектов, традиционно рассматриваемых в рамках этой дисциплины. Рассматривается формальное представление модели программно-управляемых объектов и модель класса неисправностей, на основе которых возможна автоматизация процедур построения эффективных контрольных и диагностических тестов. Оценивается сложность таких процедур и прогнозируется качество получаемых на основе этих моделей тестов.

***Ключевые слова*:** синтез тестов, диагностика**,** модель класса неисправностей, программно-управляемый объект

Введение. Программно-управляемые компоненты, используемые в современных вычислительных устройствах, являются наиболее сложными не только с точки зрения организации процессов обработки информации, но и решения задачи выбора множества воздействий, позволяющих однозначно судить об их техническом состоянии.

При синтезе тестов для таких компонент (в дальнейшем программно-управляемых объектов диагностирования – ПОД) рассматривают модели, близкие к системному уровню [1]. Главная особенность методов, ориентированных на эти модели заключается в том, что тест, также как и сам объект диагностирования (ОД), должен быть определен в терминах этого же уровня. Следовательно, к тесту как объекту проектирования, может быть применена декомпозиция, в результате которой можно выделить множество элементарных тестов и определить способ их композиции. Обычно тест для ПОД определяют как условную либо безусловную последовательность элементарных тестов, называя при этом его диагностической процедурой (ДП) [2, 3].

Вполне очевидно, что в качестве воздействий для ПОД на системном уровне должны выступать команды, определяющие преобразования, и операнды, участвующие в них.

В дальнейшем предполагается, что элементарный тест – это множество воздействий, позволяющих однозначно определить корректность выполнения той или иной команды. При этом следует уточнить, что элементарные тесты должны быть построены таким образом, чтобы их порядок следования не влиял на результаты тестирования, т.е. элементарные тесты должны быть независимыми и автономными (не требующими дополнительных воздействий на ОД до и после их реализации).

Определение теста для ПОД. В связи с этим элементарным тестом ОД будет называться пара $T =< L,\tau >,$ где $L =< l_1\ l_2\ \ldots l_i >$ – последовательность входных воздействий, $\tau = F(L, f(X))$ – реакция ОД на последовательность L. Очевидно, что тест T_i может ассоциироваться с самой командой I_i. При этом полнота этого теста (относительно предполагаемых классов неисправностей) в существенной мере зависит от качественного выбора операндов, составляющих множество L.

Если предположить, что множество элементарных тестов известно, то решение задачи синтеза тестов может быть сформулировано следующим образом:

– Синтез проверяющей процедуры (аналог контролирующего теста) заключается в выборе множества команд ПОД, корректность выполнения которых гарантирует отсутствие неисправностей из предполагаемого класса.

– Синтез диагностической процедуры предполагает получение последовательности элементарных тестов, позволяющей локализовать неисправности из заданного класса с точностью до требуемой глубины диагностирования.

Синтез диагностических процедур. В настоящее время большинство устройств микропрограммного управления ПОД реализовано на основе жесткой логики, что определяется высокими требованиями к быстродействию. Реализация функций возбуждения микрокоманды на основе жесткой логики приводит к тому, что неправильное выполнение некоторой микрокоманды в одних командах и правильное в других маловероятно. Данное обстоятельство обусловливает следующее характерное свойство ПОД.

Пусть $I_i, I_j,\ i \neq j$ команды ПОД. Тогда, в общем случае, $MP_i \cap MP_j \neq \varnothing$, где MP_i – множество микрокоманд, входящих в микропрограмму, соответствующей команде I_i, и, следовательно, $G_i \cap G_j \neq \varnothing$, где G_i, G_j – абстрактные графы выполнения (АГВ) команд.

Таким образом, предположение о реализации микропрограммного управления на основе жесткой логики и известный набор микрокоманд позволяют построить проверяющий тест. Однако для ПОД с недоступным уровнем микропрограммного управления активизация отдельных микрокоманд невозможна. В этом случае одним из способов косвенного перебора множества микрокоманд является получение подмножества команд:

$$I_n \subseteq I, \left(I_n \left(= l : \bigcup_{i=1}^{l} M_i = M, M_i \leftarrow I_i \in I_n \right) \right.$$

В случае синтеза диагностического теста для такого ОД глубина диагностирования определяется общими для последовательностей $MP_i, i = \overline{1,n}$ сегментами.

Качество диагностической процедуры может оцениваться глубиной диагностирования, вычислительной сложностью и полнотой теста относительно принятых классов неисправностей. Оптимизация диагностических процедур, соответствующих понятию полного теста, применительно к методам, рассматриваемым ниже, сводится:

- к минимизации вычислительной сложности при фиксированном значении глубины диагностирования;
- к достижению максимальной глубины диагностирования при допустимой вычислительной сложности.

Выбор множества недоминируемых команд ПОД. В дальнейшем предполагается, что для ПОД известны множество микропрограмм $\{MP_i\}$ и АГВ команд $\{G_i\}$ ($i=\overline{1,n}$, $n=|I|$).

Множество команд (элементарных тестов) $I_Д$, выбранное в качестве диагностических воздействий, должно удовлетворять условию:

$$I_i, I_j \in I_Л\,(I_i \neq I_j \rightarrow \min(MP_i \cap MP_j)) \quad \text{или} \quad I_i, I_j \in I_Л\,(I_i \neq I_j \rightarrow \min(G_i \cap G_j)).$$

В предельном случае множество $I_д$ позволяет произвести декомпозицию ОД на непересекаемые подструктуры, что соответствует выполнению условия $I_i, I_j \in I_Л\,(I_i \neq I_j \rightarrow G_i \cap G_j = \varnothing)$.

При решении задачи нахождения множества $I_д$ может быть использован подход, основанный на упорядочивании, т.е. построении отношения предпочтения, на множестве команд. Например, из двух команд $I_l, I_k, l \neq k$ предпочтительной будет I_l если выполняется условие $\forall i,j : MP_k = seg_{ij}(MP_l)$, т.е. $G_l \cap G_k = G_l$, где G_l, G_k – АГВ команд I_l, I_k. Таким образом, команда I_i структурно доминирует над командой I_j ($I_j \leq I_i$) если $G_j \subseteq G_i$.

Поскольку микропрограммы представляют собой последовательности, то в дальнейшем будет использована терминология и обозначения, приведенные ниже.

– $\langle\ \rangle$ или $\varnothing$ обозначает пустую последовательность;

– $\langle x \rangle$ обозначает последовательность, состоящую из одного элемента x;

– если $X = \langle x_1, x_2, ..., x_n \rangle$ и $Y = \langle y_1, y_2, ..., y_m \rangle$, то последовательность $C = X \cdot Y = \langle x_1, x_2, ..., x_n, y_1, y_2, ..., y_m \rangle$ является конкатенацией последовательностей X и Y;

– если $X = \langle x_1, x_2, ..., x_n \rangle$, то последовательность $C = seg_{ij}(X) = \langle x_i, x_{i+1}, ..., x_j \rangle$ ($1 \leq i \leq j \leq n$) является сегментом X;

– если $X = \langle x_1, x_2, ..., x_n \rangle$, то $first_i(X) = seg_{1i}(X)$, $rest_i(X) = seg_{in}(X)$.

Нетрудно убедиться в том, что это отношение обладает свойствами:

– рефлексивности: $\forall I_i : I_i \leq I_i$, т.к. $G_i \subseteq G_i$;

– транзитивности: $\forall I_i, I_j, I_k \in I, i \neq j \neq k : (I_i \leq I_j, I_j \leq I_k) \Rightarrow (I_i \leq I_k)$, т.к. $(G_i \subseteq G_j, G_j \subseteq G_k) \Rightarrow (G_i \subseteq G_k)$;

– антисимметричности: $\forall I_i, I_j, i \neq j : (I_i \leq I_j) \Rightarrow (I_j \leq I_i)$, т.к. из $G_i \subseteq G_j$ не следует обратное.

Указанные свойства отношения структурного доминирования позволяют охарактеризовать его как частичное упорядочение на множестве команд. Рассмотрев все пары $I_i, I_j, i \neq j$ можно построить систему множеств $S=\{O_k\}$, $k=\overline{1,|\{O_k\}|}$, где $O_k = \langle I_1 I_2 ... I_{n_k} \rangle$ – частичный порядок. Наличие множества $\{O_k\}$ обусловлено тем, что в системе команд могут встретиться такие $I_i, I_j \in \{I\}, i \neq j$, что для них не выполняется ни одно из условий $I_i \leq I_j, I_j \leq I_i$, т.е. эти команды несравнимы в смысле структурного доминирования. Очевидно, что в этом случае I_i и I_j принадлежат различным порядкам.

Поскольку каждой I_i соответствует микропрограмма MP_i и отношение структурного доминирования определено лишь для пар $I_i, I_j, i \neq j$ удовлетворяющих условию: $\exists n < m: MP_i = seg_{nm}(MP_j)$ или $MP_j = seg_{nm}(MP_i)$, то может быть сформулировано следующее утверждение.

Утверждение 1. *Если для команд $I_i, I_j, i \neq j$ имеет место $I_i \leq I_j$, то из корректного выполнения I_j следует корректность выполнения I_i.*

Множество недоминируемых команд, (для которых выполняется условие $\forall k\ \exists I_{н_к} \in O_k\colon\ \forall I_i \in O_k\colon\ I_i \le I_{нк}$), соответствующее формированию множества правых граней $\{\sup O_k\}, k = \overline{1, \langle\{O_k\}\rangle}$ (на рис. 1 это множество выделено затенением), является необходимым для построения проверяющего теста ПОД.

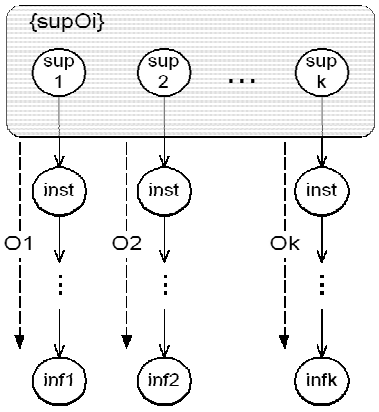

Рис. 1

Метод синтеза проверяющего теста на основе использования множества недоминируемых команд. Множество $\{\sup O_k\}, k = \overline{1, \langle\{O_k\}\rangle}$ неупорядочено, поскольку $\sup O_i$, $\sup O_j$, $(i \ne j)$ несравнимы в смысле структурного доминирования.

Наиболее простым способом получения проверяющего теста для ПОД является последовательное выполнение элементарных тестов, соответствующих командам $\sup O_k$. Однако, в этом случае вычислительная сложность диагностической процедуры может быть неприемлемо большой. Таким образом, невозможно построить эффективный проверяющий тест без введения дополнительных критериев оценки команд, позволяющих упорядочить множество недоминируемых команд.

Ниже рассмотрен подход, позволяющий синтезировать проверяющий тест, минимизированный по средней вычислительной сложности.

Пусть ОД представлен АГВ команд и найдено множество $\{\sup O_k\}$. Каждая команда I_i этого множества соответствует элементарному тесту, декомпозирующему ОД на две непересекающиеся подструктуры G_i и $\bigcup_{j=1}^{n} G_j \setminus G_i$, подозреваемые на неисправность. При этом структурная сложность идентифицируемых подструктур может быть различной.

Для характеристики сложности АГВ здесь и далее используются определенные ниже величины. Пусть $G_i = \{E_i, R_i\}$ – АГВ команды I_i, а также $\bigcup_{j=1}^{n} G_j = \{E, R\}$. Тогда планарной сложностью АГВ G_i является величина $Pl(G_i) = \dfrac{|E_i|}{|E|}$, а структурной сложностью – $St(G_i) = Pl(G_i) \cdot \dfrac{|R_i|}{|R|}$.

Естественно предположить, что априорная оценка вероятности появления отказа в какой-либо компоненте диагностируемости пропорциональна структурной сложности последней. При этом предполагается, что отказы подмножеств аппаратуры, соответствующих примитивам операционного уровня, равновероятны.

Таким образом, если для множества $\{\sup O_k\}$ определено множество то на $\{\sup O_k\}$ можно ввести отношение упорядочивания:

$$<\colon\ \forall I_i, I_j \in \{\sup O_k\}\colon\ (I_i < I_j) \to \left(St(G_i) < St(G_j)\right).$$

Введенное отношение может служить основой для построения проверяющей процедуры, соответствующей понятию проверяющего теста. Сама процедура диагностирования представлена на рис. 2.

Пусть $(T_i), i=\overline{1,n}$ – вычислительная сложность элементарного теста T_i. Тогда средняя вычислительная сложность проверяющей процедуры, изображенной на рис. 2, может быть оценена величиной: $C_{\text{ДП}} = \dfrac{\sum_{i=1}^{n} C(T_i) \cdot \prod_{j=1}^{i} St(G_i)}{n+1}$.

Введение отношений $<$, $\le$ на множестве команд ПОД оказывается полезным также и при построении минимизированных по вычислительной сложности диагностических с точностью до определенной глубины процедур.

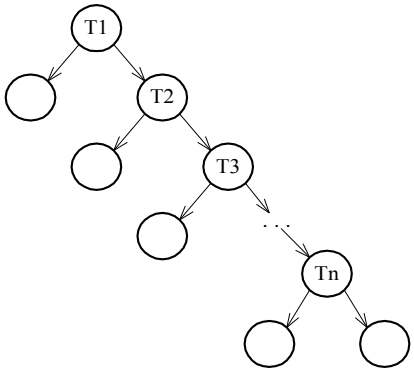

Рис. 2.

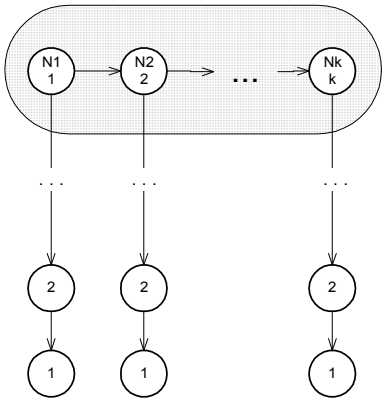

Рис. 3

Пусть в соответствие множеству $\{T_i\}$, $i=\overline{1,n}$, где T_i – элементарный тест для команды I_i, поставлено множество вершин графа T, множество ребер R определяет отношения $<$ и $\le$. Тогда $D=\{T,R\}$ – есть диагностическая процедура, определенная как отношение следования на множестве элементарных тестов.

Для $D=\{T,R\}$, изображенного на рис. 3, введена следующая нумерация вершин: $\forall i$: $\inf O_i \to 1$, $\sup O_i \to Ni = |O_i|$. Для вершин, соответствующих $\sup O_j$, $j=\overline{1,k}$ введена еще одна нумерация: $\sup O_0 \to 1$, $\inf O_0 \to k$. Как видно из рис. 3, правая ветвь процедуры диагностирования является проверяющим тестом ПОД.

Поскольку для команд, входящих в порядок O_i справедливо утверждение 1, то локализацию неисправности (здесь имеется ввиду граф $\gamma = G_{i+1} \setminus G_i$) можно производить с помощью метода половинного деления. В этом случае вычислительная сложность составляет величину $O\left(\log_2(|O_i|)\right)$, что свидетельствует о достаточной эффективности диагностических процедур, синтезируемых в соответствии с изложенным подходом.

Следует отметить, что такие процедуры целесообразны в случае обеспечения ими требуемой глубины диагностирования, поскольку компоненты диагностируемости, с точностью до которых можно производить локализацию неисправности, определены как $\gamma = G_{i+1} \setminus G_i$.

Использование функции предпочтения в методах синтеза диагностических процедур. Как отмечалось раннее, наличие множества недоминируемых команд ПОД является необходимым (но не достаточным) условием построения проверяющего теста, что обусловливает его возможную избыточность. Такая избыточность вызвана ограничениями, связанными со сравнимостью команд в смысле структурного доминирования.

В случае ослабления условия сравнимости команд можно получить множество $I_{\text{д}} \subseteq \{\sup O_k\}$, достаточное для построения проверяющего теста.

Пусть команды $I_k, I_m \in I$ сравнимы, если

$$\exists i,j,l,k \quad \text{и} \quad S\copyright: S\copyright = seg_{ij}(MP_n), S\copyright = seg_{lk}(MP_m), \tag{1}$$

где MP_n, MP_m – микропрограммы команд I_n, I_m.

Для того, чтобы упорядочить сравнимые в смысле (1) команды, необходимо ввести некоторый критерий предпочтения, определенный, как минимум, на парах команд:

$$\Pr: I_i, I_j \to f(i,j), f \in R. \tag{2}$$

Как уже упоминалось, в случае предположения о равновероятности отказов аппаратных средств, соответствующих примитивам операционного уровня, критерием «полезности» команды является структурная сложность соответствующего ей АГВ. Если рассмотреть команды I_i и I_j, то в зависимости от корректности их выполнения подозреваться на неисправность будет одна из структур: $G_i \cap G_j$, $G_i \setminus G_j$, $G_j \setminus G_i$, $G \setminus (G_i \cup G_j)$.

Соответствующие им структурные сложности можно использовать в качестве частных критериев предпочтения:

$$K_{ij}^1 = St(G_i) + St(G_j); \quad K_{ij}^2 = St(G_i \cap G_j); \quad K_{ij}^3 = St(G_i \setminus G_j).$$

Так как эти критерии являются монотонными и выполняется условие: $\langle I_i, I_j \rangle << \langle I_i, I_l \rangle: K_{ij}^k < K_{il}^k$ и $\exists r: K_{ij}^r < K_{il}^r (k = \overline{1,3})$, то можно определить функцию: $f(i,j) = K_{ij}^1 + K_{ij}^2 + K_{ij}^3$, которая удовлетворяет условиям выражения (2). В этом случае задача синтеза проверяющей процедуры может быть сформулирована как многошаговая задача принятия решений (выбора команды) при условии упорядочения пар альтернатив. Т.е. если имеется множество $\{T_i\}$ элементарных тестов, причем $T_i \to I_i (I_i \in I)$, то на *m*-ом шаге синтеза проверяющего теста необходимо выбрать T_{m+1}, который удовлетворяет условию $T_{m+1}=T_i: \max_i(f(m,i))$.

При этом *m+1* шаг является последним, если

$$\{T_l\} \to \{I_l\}, \; I_l \to G_l \;\; (l = \overline{1,m+1}), G_\partial = \bigcup_{l=1}^{m+1} G_l: \; \forall G_i \;\; (i = \overline{1,n}): \; G_i \subseteq \bigcup_{j=1}^{m+1} G_j,$$

что соответствует условию полноты проверяющей процедуры.

В дальнейшем рассматривается подход к синтезу ДП.

Пусть ДП представляет собой условный порядок, т.е. $T_i \xrightarrow{D} T_l(T_i), T_r(T_i)$, причем отношение следования определяется как $D = \begin{cases} \langle T_i, T_r(T_i) \rangle, \text{если } T_i \text{ выполнен корректно}; \\ \langle T_i, T_l(T_i) \rangle, \text{если } T_i \text{ выполнен некорректно}. \end{cases}$

Это отношение можно интерпретировать бинарным деревом, изображенным на рис. 4.

Листья такого дерева являются исходами, соответствующими компонентам диагностируемости, а нетерминальные вершины – элементарными тестами. Необходимо отметить, что последовательность $D_r = \langle T_1, ..., T_r(T_r(...T_r(T_1)...)) \rangle$ (на рис. 4. выделена затенением) представляет собой проверяющий тест, построение которого рассматривалось выше.

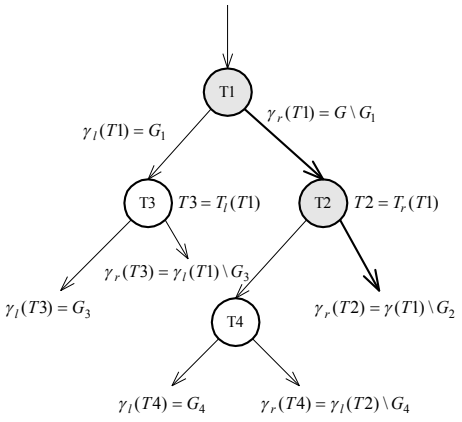

Рис. 4

Задача синтеза ДП может быть сведена к задаче выбора на *i*-ом шаге пар $\langle T_i, T_l(T_i) \rangle, \langle T_i, T_r(T_i) \rangle$, удовлетворяющих условию:

$$T_m = T_r(T_i), T_n = T_l(T_i): \max_{n,m}(f_r(i,m) + f_l(i,n)), \tag{3}$$

где $f_r(i,m), f_l(i,n)$ – функции предпочтения для правой и левой ветви.

Пусть в ОД существует одиночная функциональная неисправность, тогда $G = \bigcup_{i=1}^{n} G_i$ (G_i – АГВ команды I_i, а n определяет мощность множества команд) подозревается на неисправность. Тест T_1 разбивает универсум G на две подструктуры $\gamma_l(T_1) = G_1, \gamma_r(T_1) = G \setminus G_1$, одна из которых может содержать неисправность.

Выбор $T_l(T_1)$ должен осуществляться в соответствии с условием:

$$T_i = T_l(T_1) : G_i \cap \gamma_l(T_1) \neq \varnothing . \quad (4)$$

Очевидно, что существует множество команд, для которых выполняется условие (4). В связи с этим предпочтительно выбирать ту команду, которая обеспечит разбиение подозреваемой на неисправность структуры на две части, примерно равные по объему. Это обозначает, что функция $f\copyright_l(1,i) = \left\langle St(G_i) - St(G_1 \setminus G_i) \right\rangle$ принимает минимальное значение для наиболее предпочтительной альтернативы на шаге 1.

Для того, чтобы обобщить эту функцию на произвольный шаг, необходимо определить для любого элементарного теста T_i, входящего в ДП:

$$\gamma_l(T_i) = G_i , \qquad \gamma_p(T_i) = \begin{cases} \gamma_l(T), & T_i = T_l(T); \\ \gamma_r(T), & T_i = T_r(T). \end{cases} \qquad \gamma_r(T_i) = \gamma_p(T_i) \setminus G_i .$$

Учитывая то, что согласно выражению (3) целевая функция должна иметь максимальное значение для наилучшей альтернативы, окончательное выражение для нее имеет вид:

$$f_l(i,j) = \frac{1}{\left\langle St\left(\gamma_l(T_j)\right) - St\left(\gamma_r(T_j)\right)\right\rangle} .$$

Применение такой функции предпочтения обеспечивает приближение структуры ДП к сбалансированному бинарному дереву, а ее вычислительной сложности – к величине $O(\log_2 n)$, где n – мощность множества элементарных тестов.

Аналогичный подход используется при определении $f_r(i,j)$.

Поскольку правые ветви ДП являются продолжением проверяющего теста, то естественно в качестве $f_r(i,j)$ выбрать функцию $f_r(i,j) = St(G_j)$.

Описанный подход к синтезу диагностических процедур сочетает полноту теста, соответствующего D_r, и эффективную идентификацию компонент диагностируемости, характеризующих глубину, близкую к максимально допустимой.

Минимизация затрат памяти для хранения диагностических процедур. Объем памяти, необходимый для хранения диагностических процедур (ДП), оказывает значительное влияние на характеристики устройства диагностирования в целом.

В связи с этим может быть поставлена задача минимизации затрат памяти для представления ДП при сохранении ими таких свойств как глубина диагностирования и средняя вычислительная сложность.

Диагностическая процедура может быть описана как дискретная функция $D(T)$, определенная на множестве всех двоичных наборов $T = \langle t_1, t_2, ..., t_n \rangle$ (t_i соответствует корректности выполнения теста T_i). Область значений этой функции соответствует множеству компонент диагностирования Г. Функция $D(T)$ в дальнейшем будет называться функцией диагностирования.

Функцию диагностирования можно интерпретировать бинарным деревом, которое в дальнейшем будет называться моделью диагностической процедуры (МДП).

МДП представляет собой бинарное дерево, в котором: – каждой нетерминальной вершине соответствует переменная t_i; – каждой терминальной вершине соответствует элемент множества Γ_D; – каждое ребро взвешено значениями переменных t_i.

Полная МДП представляет собой полное сбалансированное бинарное дерево.

Любая вершина этого дерева является корнем некоторого поддерева. При этом $T_R(i),\ T_L(i)$ есть правое и левое поддерево для вершины i. Вершины $N_R(i)$ ($N_L(i)$) есть корень $T_R(i)$ ($T_L(i)$). Вершина $P(i)$ является предшественником вершины i. Для определенности введено правило: $\forall i \in V_H\ \ C_{i,N_R(i)} = 1,\ C_{i,N_L(i)} = 0$.

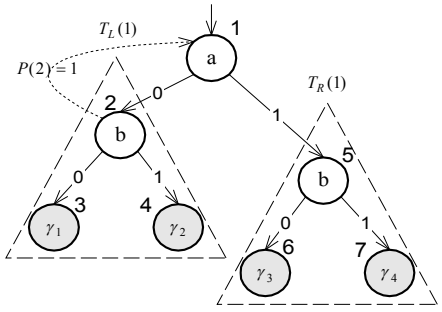

Рис. 5.

На рис. 5. приведен пример МДП.

Каждый из путей, ведущих из корня МДП в одну из терминальных вершин, соответствует результатам проведения диагностического эксперимента. Так как определение элементарного теста предполагает, что корректность его выполнения не зависит от последовательности выполненных до этого тестов и не влияет на выполнение других, то и порядок следования вершин в пути не влияет на идентификацию компоненты диагностируемости, поставленной ему в соответствие. Это означает, что неупорядоченное множество пар $t_k = \left\{\left\langle C_i, C_{i,j}\right\rangle\right\}$, где C_i – вес вершины i, $C_{i,j}$ – вес ребра $\langle i, j\rangle$, вершины i, j входят в один путь и $P(j) = i$, однозначно определяет исход диагностического эксперимента, а $T = \{t_k\}\ (k = \overline{1,m}$, m соответствует числу терминальных вершин МДП) – все исходы тестового эксперимента.

В дальнейшем множество t_i будет называться набором, а соответствующая ему компонента диагностируемости – значением функции диагностирования на этом наборе, т.е. $f(t_i) = \gamma_i$. Если для наборов t_1 и t_2 имеет место: $t_1 \subseteq t_2$ и $f(t_1) = f(t_2)$, то это означает, что пары из $t_2 \setminus t_1$ не влияют на результат диагностического эксперимента, т.е. набор t_1 «покрывает» набор t_2.

Метод минимизации диагностических процедур. Пусть для ДП построена полная МДП. Необходимо минимизировать число вершин МДП при условии сохранения диагностической процедурой глубины диагностирования и средней вычислительной сложности. Для этого введем отношение частичного порядка для множества МДП:

$$G_1 \subseteq G_2,\ \text{если}\ \exists\ t_i \in G_1,\ \ t_j \in G_2 \colon\ t_i \subseteq t_j\ \ \text{и}\ \ f_1(t_i) = f_2(t_j), \qquad (5)$$

которое обладает свойствами: рефлексивности – $(G_1 \subseteq G_1)$; антисимметричности – $(G_1 \subseteq G_2) \Rightarrow (G_2 \subseteq G_1)$; транзитивности – $(G_1 \subseteq G_2, G_2 \subseteq G_3) \Rightarrow (G_1 \subseteq G_2)$.

Частным случаем отношения частичного порядка является отношение эквивалентности: $G_1 \cong G_2,,$ если $\exists\ t_i \in G_1,\ t_j \in G_2 \colon\ t_i = t_j, f_1(t_i) = f_2(t_j)$, которое обладает свойствами рефлексивности, симметричности и транзитивности.

Теорема 1.

Если $G(f_1) \subseteq G(f_2)$, то $G(f_1)$ и $G(f_2)$ описывают одну и ту же функцию.

Доказательство.

Исходя из определения (5) частичного порядка для МДП следует, что для каждого набора t_2 МДП $G(f_2)$ в МДП $G(f_1)$ найдется набор t_1, который покрывает t_2, что, в свою очередь, означает $f_1 = f_2$.

Следствие 1.1.

Если существует частичный порядок $G(f_1)G(f_2)\ldots G(f_n)$ *и* $G(f_1)$ *– левая грань частичного порядка, то* $G(f_1)$ *есть минимальная МДП для функции* $f = f_n = \ldots = f_1$.

Доказательство.

Минимальная МДП – есть МДП, содержащая минимальное число вершин. Последовательное применение теоремы 1 к парам G_{i-1}, $G_i, i = \overline{n,2}$ и то, что $G(f_1)$ – левая грань частичного порядка, приводит к выводу о том, что $G(f_1)$ является минимальной МДП для функции $f = f_n = \ldots = f_1$.

Следствие 1.2.

Если $G(f_1) \cong G(f_2)$, *то* $f_1 = f_2$.

Доказательство очевидно. Следует отметить, что $G(f_1)$ и $G(f_2)$ имеют одинаковое число вершин.

Теорема 2.

Если $\exists\, i \in G(f)$: $T_R(i) \cong T_L(i) = T$, *то* G_2, *в которой отсутствует вершина* i *и*

$$\begin{cases} T_R(P(i)) = T, \text{ если } i = N_R(P(i)) \\ T_L(P(i)) = T, \text{ если } i = N_L(P(i)) \end{cases},$$

удовлетворяет отношению $G_2(f) \subseteq G(f)$.

На рис. 6. приведен пример такого преобразования.

Теорема 3.

Если $\exists i \in G(f)$: $C_{N_R(i)} = C_{N_L(i)}$, *то* $G_2(f)$, *в которой:* $C_i^2 = C_{N_R(i)} = C_{N_L(i)}$, $C_{N_R(i)}^2 = C_{N_L(i)}^2 = C_i$, $T_R^2(N_R(i)) = T_R(N_R(i))$, $T_L^2(N_R(i)) = T_R(N_L(i))$, $T_R^2(N_L(i)) = T_L(N_R(i))$, $T_L^2(N_L(i)) = T_L(N_L(i))$, *удовлетворяет отношению* $G_2(f) \cong G(f)$.

На рис. 7. приведен пример такого преобразования.

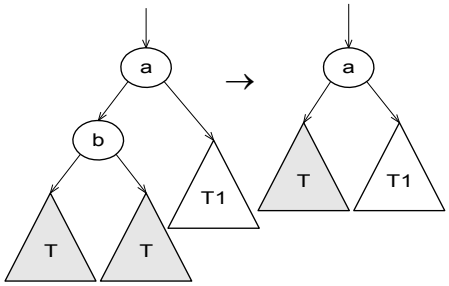

Рис. 6

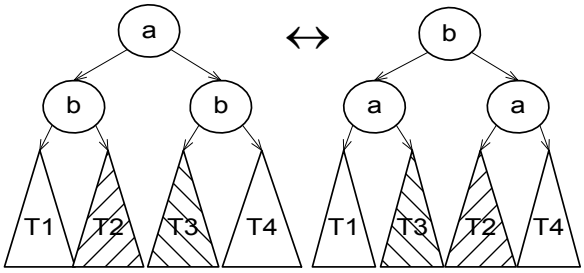

Рис. 7

Следует заметить, что для реализации алгоритма минимизации МДП достаточно реализовать преобразования, описанные в теореме 2. Однако алгоритм поиска эквивалентных деревьев с произвольным числом вершин обладает большой вычислительной сложностью. В связи с этим проще всего этот алгоритм реализуется проще всего для деревьев, содержащих одну (терминальную) вершину. Если при этом использовать рекурсивный алгоритм обхода бинарного дерева «в глубину», то он позволит произвести за один проход МДП все возможные преобразования частичного порядка. Недостаток этого алгоритма состоит в том, что результирующая МДП не является минимальной. Следовательно, необходимо также реализовать преобразования, описанные в теореме 3, после чего повторить снова преобразование частичного порядка.

Следует отметить, что данный метод не снижает глубины диагностирования МДП поскольку при описанных преобразованиях область значений функции диагностирования не изменяется.

Пусть $\langle T \rangle = n$, где T – множество аргументов функции диагностирования. Тогда средняя вычислительная сложность МДП составляет величину $t_1 = \frac{2^n n}{2^n} = n$.

Если при минимизации МДП сокращено m вершин, то в наихудшем случае среднее время диагностирования для минимизированной ДП определяется как $t_2 = \frac{(2^n - m)n}{2^n} = n - \frac{n \cdot m}{2^n}$.

При достаточно больших n $t_2 = n$, т.е. рассмотренный метод минимизации затрат памяти для представления ДП удовлетворяет предъявляемым требованиям.

Заключение. В работе сформулирован подход к построению моделей и преобразований изображений, который отличается от известных тем, что учитывает семантику последних. Это позволяет получать формализованные математические схемы решения задач анализа изображений с учетом их семантики. Формализация математической схемы решения задачи анализа изображений создает предпосылки к созданию унифицированного программного каркаса, использование которого существенно снижает затраты на разработку специализированных программных средств связанных с анализом изображений.

В качестве примера применения сформулированного подхода приведено решение задачи анализа изображений схем в технологических процессах автоматизированного проектирования.

Литература

1. Бельхадри Н.Обеспечение надежности компьютерных систем / Н. Бельхадри, А.А. Болдак, Т.М. Чебаненко ; под ред. О.В. Бузовского. – Київ : ТОО «ВЕК», 2007. – 160 с.

2. Андерсон Р.Н. Методы контроля микропроцессорных устройств / Р.Н. Андерсон // Электроника. – 1996. – No 8. – С. 82-89.

3. Малышенко Ю.А. Автоматизация диагностирования электронных устройств / Ю.А. Малышенко, В.В. Чипулис, С.Г. Шаршунов ; под ред. В.П.Чипулиса. – Москва : Энергоатомиздат, 1986. – 216 с.

Автори статті

Бузовський Олег Володимирович – доктор технічних наук, професор кафедри обчислювальної техніки, Національный технічний університет України «Київський політехнічний інститут», Київ. Тел. +380 (67) 185 11 85. E-mail: obuza@i.ua

Чебаненко Тетяна Михайлівна – кандидат технічних наук, доцент кафедри обчислювальної техніки, Національный технічний університет України «Київський політехнічний інститут», Київ. Тел. +380 (67) 185 11 85. E-mail: obuza@i.ua

Authors of the article

Buzovskyy Oleh Volodymyrovich – doctor of science (technical), professor of the computer engineering department, National Technical University of Ukraine "Kyiv Polytechnic Institute", Kiev. Tel.: +380 (67) 185 11 85. E-mail: obuza@i.ua

Chebanenko Tetyana Mykhaylivna – candidate of science (technical), associate professor of the computer engineering department, National Technical University of Ukraine "Kyiv Polytechnic Institute", Kiev. Tel.: +380 (67) 185 11 85. E-mail: obuza@i.ua

Дата надходження в редакцію: 29.02.2016 р. Рецензент: д.т.н., проф. В.П. Тарасенко