УДК 658.512.011:681.326:519.713

ИНФРАСТРУКТУРА АНАЛИЗА И ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ КИБЕРПРОСТРАНСТВА

ХАХАНОВ В.И., ЧУМАЧЕНКО С.В., ЛИТВИНОВА Е.И., МИЩЕНКО А.С., АДАМОВ А.С.

Предлагается модель и метрика кибернетического пространства, где субъектами выступают взаимодействующие процессы или явления с физическим носителем в виде компьютерных систем и сетей. Разрабатывается оценка меры бинарного отношения двух объектов в киберпространстве для распознавания любого типа взаимодействия объектов при решении практических задач информационной безопасности. Предлагается модель быстродействующего гетерогенного мультиматричного процессора, предназначенного для быстрого поиска информационных объектов в киберпространстве. Описываются инфраструктуры сервисного обслуживания программного продукта для защиты от вредоносных программ и верификации проектов цифровых систем. Предлагается структурная модель распознавания деструктивных образов на основе отношений входного образа, эталонов и критериев сходства.

1. Введение

Киберпространство (cyberspace) – метафорическая - абстракция, используемая в философии и в компьютерах, – является виртуальной реальностью, которая представляет Ноосферу или Второй мир «внутри» компьютеров и сетей. Это определение взято из Википедии и согласуется с многочисленными публикациями, авторами которых в абсолютном большинстве являются философы. Возможно, уже настало время взглянуть на определение и сущность киберпространства с позиции технологической и математической культуры. Основная аксиома, заложенная в определении, виртуальность пространства.

Виртуальность (virtus – потенциальный, возможный) – вымышленный, воображаемый объект или явление,

не присутствующее в реальном мире, а созданное игрой воображения человеческой мысли, либо смоделированное при помощи других объектов. Реальные компьютеры создают «нереальное пространство», а точнее, Второй мир, который вполне может стать первым. Не хватает малого – целенаправленного самосовершенствования компьютерного сообщества, способного на равных с человеком участвовать в создании и формировании киберпространства. В нем информация – реальность и сущность пространства, компьютер – ее носитель или форма.

Киберпространство – совокупность взаимодействующих по метрике информационных процессов и явлений, использующих в качестве носителя компьютерные системы и сети.

Метрика – способ измерения расстояния в пространстве между компонентами процессов или явлений. Расстояние в киберпространстве это – кодовое расстояние по Хэммингу между парой векторов, обозначающих компоненты процесса или явления. Расстояние, производная (булева), степень изменения, различия или близости есть изоморфные понятия, связанные с определением отношения двух компонентов процесса или явления. Понятие близости (расстояния) компонентов в киберпространстве есть мера их различия. Производная – мера бинарного отношения динамических или статических компонентов в процессах или явлениях. Процедуры сравнения, измерения, оценивания, распознавания, тестирования, диагностирования, идентифицирования имеют место при наличии хотя бы одного отношения.

Рыночная привлекательность киберпространства. Тенденция последних лет в части создания новых коммуникационных, вычислительных и информационных сервисов, полезных для человека, обращает внимание на создание все более специализированных гаджетов (gadget), имеющих существенные преимущества перед персональными компьютерами и ноутбуками: энергопотребление, компактность, вес, стоимость, функциональные возможности, дружественность интерфейса. Практически вся десятка лучших за 2010 год специализированных изделий (Apple iPad, Samsung Galaxy S, Apple MacBook Air, Logitech Revue, Google Nexus One (HTC Desire), Apple iPhone 4, Apple TV, Toshiba Libretto W100, Microsoft Kinect, Nook Color) реализована в виде цифровых систем на кристаллах. К 2012 году рынок мобильной и беспроводной связи перейдет на 20 нм (итоги январского Саммита 2011 альянса Common Platform). Дальнейшее развитие технологий по годам: 2014 – 14 нм, 2016 – 11 нм. В 2015 году 55% сотовых телефонов станут смартфонами, планшетные компьютеры заменят ноутбуки и нетбуки. Суперфоны (Nexus-1, Google) станут той соединительной тканью, которая свяжет все остальные устройства и сервисы. Переход от вычислительных платформ к мобильным устройствам с малым форм-фактором приведет к существенному снижению энергопотребления во всем мире. Надвигается следующая волна компьютеризации под названием «интернет вещей», которая приведет к широкому распространению датчиковых сетей, включая их интеграцию в человеческое тело. Мировой рынок перечисленных выше устройств и гаджетов насчитывает сегодня порядка 3 миллиардов изделий.

С учетом изложенного выше можно сделать следующие выводы в отношении эволюционирования киберпространства: 1) Персональный компьютер, уходя с рынка, трансформируется в широкий спектр гаджетов доступа в киберпространство, обладающих функциональностью персональных компьютеров, компактностью и низкой стоимостью. 2) Как интерфейс связи между человеком и киберпространством гаджет в меньшей степени нуждается в защите. 3) Киберпространство имеет иерархию от индивидуального дружественного пространства пользователя до глобального, где фигурируют «облака», данные и сети по интересам. 4) Для повышения информационной безопасности предметного (целевого) киберпространства возможно делать его блуждающим, что затруднит определение его точного адреса. 5) Нарождается новая структура в виде киберпространства как части общей экосистемы планеты, для которой необходимо создавать инфраструктуру информационной безопасности как средство сервисного обслуживания. 6) Стремительно развивается мощный сегмент рынка планшетных компьютеров, которые не имеют входов и выходов, кроме Internet. Такая ситуация подталкивает пользователя к созданию индивидуального киберпространства, независимого от типа гаджета, которое должно быть надежно защищено. 7) Сегодня еще высокая стоимость программных приложений на компьютере экономически оправдывает применение антивирусных защит. Но завтра, для дешевых устройств типа, IPad и IPhon, с низкой мощностью программных приложений антивирусная защита станет экономически нецелесообразной. Одновременно возникает новый субъект для защиты – индивидуальное киберпространство (данные и приложения), которое получает сервисы от технологических «облаков» и сетей по интересам. 8) Одним из возможных решений для охраны пространства может быть его вакцинация в виде введения некоторой избыточности, которая дает возможность осуществлять мониторинг и экстренную связь с облаком антивирусных сервисов в целях удаления функциональных нарушений. 9) Вакцинация данных и программ на персональном компьютере также может быть интересным решением. Она позволяет более оперативно осуществлять мониторинг данных, используя информацию от внедренных в папки и программы агентов в виде ассерционных операторов.

Один из возможных шагов в данном направлении представлен ниже в виде структуры публикации как технологии тестирования и диагностирования вредоносных компонентов (ВК – malware) $T^v : M^t$ – метрика и модель процессов тестирования, H^c – полезная функциональность (программа), G^t – граф транзакций программных блоков, $\{M^f, M^s\}$ две модели тес-

тирования программы (таблица вредоносов и матрица активизации программных блоков), $\{D^c, D^r, D^m\}$ – три метода (анализ строк, столбцов таблицы и матрицы в целом) диагностирования ВК, использующих механизм ассерций (ассерция – логическое высказывание, определяющее функциональные ошибки программного кода), P^m – создание архитектуры мультиматричного процессора для параллельного и быстрого анализа табличных данных, R – имплементация моделей методов и средств в систему – инфраструктуру информационной безопасности киберпространства (И-ИБК):

$$T^v = M^t \to H^c \to G^t \to \left\{ \begin{array}{l} M^f \to \left\{ \begin{array}{l} D^c \\ D^r \end{array} \right. \\ M^s \to D^m \end{array} \right\} \to P^m \to R.$$

Сущность исследования – существенное уменьшение времени диагностирования ВК и повышение качества сервисов киберпространства благодаря разработке ассерционно-ориентированной инфраструктуры, моделей и методов тестирования и диагностирования программных продуктов и данных. Информация, необходимая для поиска блоков с ВК, определяется в процессе моделирования (эмуляции) программного кода. Эффективность предметного киберпространства (программного изделия) определяется нормированным в интервале [0,1] интегральным критерием:

$$E = F(L, T, H) = \min[\frac{1}{3}(L + T + H)],$$

$$Y = (1 - P)^n;$$

$$L = 1 - Y^{(1-k)} = 1 - (1 - P)^{n(1-k)};$$

$$T = [(1 - k) \times H^s]/(H^s + H^a); \; H = H^a/(H^s + H^a).$$

Здесь представлены: уровень заражения вредоносными компонентами L, время тестирования T, программно-аппаратная избыточность, определяемая механизмами ассерций и средствами сервисного обслуживания H. Параметр L, как дополнение к параметру Y, характеризующему выход годной продукции, зависит от тестопригодности проекта k, вероятности P существования ВК и числа необнаруженных вредоносов n. Время тестирования определяется тестопригодностью проекта k, умноженной на структурную сложность аппаратно-программной функциональности, отнесенной к общей сложности продукта в строках кода. Программно-аппаратная избыточность находится в функциональной зависимости от сложности ассерционного кода и других избыточностей, отнесенных к общей сложности продукта. При этом программная или аппаратная, избыточность должна обеспечивать заданную глубину диагностирования деструктивов функциональности за время выхода изделия на рынок, определенное заказчиком.

Цель исследования – существенное повышение качества сервисов, доставляемых со стороны программных, аппаратных изделий, сетевых структур киберпространства и уменьшение стоимости эксплуатационных расходов за счет создания инфраструктур сервисного обслуживания и безопасности, обеспечивающих дружественную эксплуатацию, тестирование и устранение нефункциональных деструктивных компонентов.

Задачи: 1) Разработка модели кибернетического пространства. 2) Математический аппарат и двигатель для анализа и сервисного обслуживания киберпространства. 3) Процесс-модели и критерии взаимодействия вредоносных компонентов с программными кодами полезных функциональностей. 4) Инфраструктура киберпространства и реализация ее компонентов.

Источники: 1. Актуальные проблемы анализа киберпространства [1-3]. 2. Метрика киберпространства [4, 5]. 3. Аппаратура и матричные процессоры для поиска информации [6-13, 14-17, 18-28]. 4. Распознавание деструктивных образов при защите киберпространства [4,5,28].

2. Киберпространство или интеллектуальная компьютерная экосистема

Инфраструктура кибернетического пространства, метрика его измерения и процесс-модели анализа и синтеза субъектов дают возможность создавать эффективные решения компьютерных средств и технологий, ориентированных на быстрый поиск, распознавание, диагностирование не только позитивных, но и негативных субъектов. Конкретно, предложенная инфраструктура может решать задачи: 1) Описание многообразия деструктивных компонентов кибернетического пространства. 2) Формализация процессов взаимодействия триады компонентов <программа, деструктивности, тесты>. 3) Диагностирование и устранение деструктивных компонентов. 4) Создание и эффективное использование базы деструктивных данных. 5) Создание быстродействующих интеллектуальных саморазвивающихся средств сервисного обслуживания и защиты кибернетического пространства.

Сегодня чрезвычайно важно обозначить возможные пути для решения проблемы создания саморазвивающейся компьютерной экосистемы. Эволюция ИКЭС основывается на использовании трех наиболее важных компонентов: фантазия, математика и технология, где субъектом экосистемы выступает саморазвивающийся компьютер (СРК). При этом дефицит в мире фантазеров или фантастов, связанный с шаблонами обучения и постепенным закрепощением будущего инженера по мере получения им образовательных услуг в школе и университете, существенно тормозит эволюцию СРК. Основное отличие СРК от современного компьютера заключается в концепции жизненного цикла. Стратегия настоящего компьютера есть обучение или повторение уже пройденного пути. Принципиальная позиция СРК – постоянный поиск новых путей для самосовершенствования (эволюционирования) на основе мирового опыта, скрытого в инфор-

мационном пространстве. Согласно запрету Геделя, адаптированному для информационного пространства, нельзя создать компьютер, который способен решать любые задачи, формально представленные спецификацией. Тем не менее, принцип Геделя предоставляет методологическую основу эволюции (саморазвития) ИКЭС, которую можно интерпретировать следующим образом. Для информационного пространства всегда можно придумать такую полезную спецификацию, которая не покрывается существующими у человечества решениями, что обусловливает создание нового функционального или технологического компонента для его последующего включения в планетарную библиотеку. ИКЭС имеет возможность повторить эволюцию человечества, только в тысячи раз более быстрыми темпами. На рис. 1 представлен замкнутый цикл эволюции ИКЭС, который фактически изоморфен спирали развития человечества, накрученной на временную ось.

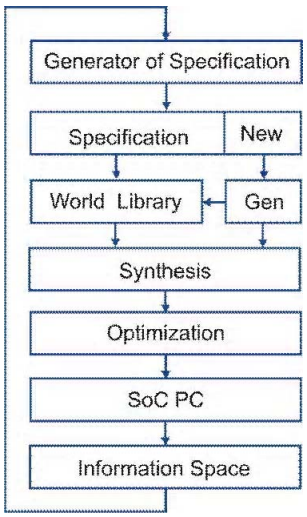


Рис.1. Цикл ИКЭС

Здесь заложены основные принципы эволюции, явно выраженные уже в современной компьютерной индустрии: 1) Стандартизация – самое главное для эволюции и жизненного цикла СРК – рынок не принимает и не понимает нестандартных по интерфейсу решений. 2) Специализация есть повышение эффективности предоставляемых (персонально ориентированных) сервисов изделия, связанных с быстродействием, качеством, затратами, энергосбережением путем оптимизации структуры и функциональных компонентов, покрывающих спецификацию. 3) Повсеместное использование векторно-логического критерия качества решения в задачах генерирования идей, синтеза и анализа. Генерирование – процесс создания новой функциональности. При этом синтез оперирует существующими в информационном пространстве компонентами для создания структуры. Анализ – оценивание полученного решения. 4) Диаграмма Хассе используется для выработки стратегии оптимизации покрытия функциональностей спецификации библиотечными компонентами или их сочетаниями, принадлежащими информационному пространству. Она согласуется с современной Y-технологией, входящей в состав ESL Design, суть которой – использовать библиотечные компоненты на всех уровнях проектирования изделия для покрытия специфицируемой функциональности в процессе синтеза.

Человек окружен природой, состоящей из многих полезных для него субъектов: флора, фауна, которые эволюционируют по своим законам. Никто не пытается сделать, например, из лошади или дельфина человека. Но в части развития компьютеров мы горим желанием сделать его мыслящим, наделить органами чувств – сотворить его подобным, в конце концов, человеку. Нужно ли это человеку? Вопрос – спорный. Не лучше ли будет взять на вооружение другой тезис – компьютер, как реальная часть природы, имеет право занимать в ней собственную нишу для самостоятельного эволюционирования и самосовершенствования. Насколько он будет подобен человеку в будущем? Трудно судить. Но он есть и будет более совершенным в одних задачах, менее – в других, также как и природные субъекты по отдельным параметрам превосходят человека. Не обижается же он на рыб, которые могут нырять и обходиться без воздуха, или на птиц, наделенных возможностью летать. Но, заимствуя у субъектов природы отдельные функциональности, человек создает изделия, расширяющие его возможности. Поэтому не следует сталкивать лбами человека и компьютер, которые и в будущем будут жить в мире и согласии, как субъекты природы, использующие друг друга для решения общественно полезных задач. Не следует также пытаться копировать живой мозг человека, по сути, в неживом кремниевом кристалле. Копия всегда будет хуже оригинала! Но инкапсулировать функциональности мозга в кристалл – значит расширить мощность сервисов, полезных для решения практических задач в информационном пространстве. Идея самостоятельного развития компьютера по собственному пути немного подрезает крылья у фантазии – сделать его неотличимым от человека. Но при этом она окрыляет практических исследователей и IT-индустрию. Становится возможным уже в ближайшее десятилетие сотворить саморазвивающуюся информационно-компьютерную экосистему! Кроме того, от взаимодействия Software и Hardware, освященного оригинальной идеей, действительно появляются полезные функциональности: специализированные изделия, мобильные телефоны, RFID, планшеты, серверы, сети, формирующие в совокупности информационное пространство планеты. Рынок, как и природа, осуществляет селекцию или естественный отбор практически ориентированных разработок, среди которых нет места слабым субъектам или проектам, а выживают сильнейшие представители в своем экологическом слое.

Относительно интеллекта. Википедия дает краткое и достаточно полное определение: «Интеллект – способность системы создавать в ходе самообучения программы для решения задач». Это относится к естественному и искусственному интеллекту. Как

следствие можно предположить, что есть нечто выше интеллекта, поскольку в приведенном определении нет места для креативной эволюции субъекта. Здесь есть чистое повторение пройденного кем-то пути с использованием обучения или самообучения. Получается, человек – только повторитель? Где же место в интеллекте для создателей новой культуры, математической и технологической, машин, строений? Определение также противоречит принципу Геделя и не оставляет места для человека как творца. Все отдается Богу, который создал этот мир, а человеку оставил только функциональность познавать его. Но ведь вселенная не стоит на месте – она развивается, можно предположить, что саморазвивается. То же самое относится и к человеку. В определении интеллекта заложен процесс познания мира. Но Творец не создавал компьютер, машины, здания, мосты. Кто же это сделал? Человек, наделенный интеллектом, что не есть только обучение для познания мира. Маленькая корректировка может существенно поправить понятие интеллекта, если в нем появятся два вида деятельности: повторение (познание) и созидание. Идея оригинальности может быть представлена как булева производная от функции-спецификации f по всем переменным-компонентам, которые должны покрыть спецификацию:

$$E = \frac{df}{dx_i} = 1 \rightarrow f(x_1, x_2, ..., x_i, ..., x_n) \oplus f(x_1, x_2, ..., \bar{x}_i, ..., x_n) = 1.$$

Если существует компонент x_i, которого нет в библиотеке решений, то производная укажет на него, что означает – необходимо создание новой, еще не известной функциональности. Формально уравнение булевой производной определяет существенность компонента x_i для реализации спецификации при его наличии или отсутствии. Естественно, если критерий качества покрытия спецификации или идеи существующим опытом человечества равен нулю, то данный факт следует понимать как повторение уже пройденного кем-то пути, если при этом не создается новая структура, которая может быть представлена одной из переменных. В противном случае фиксируется создание нового компонента или структуры, которые включаются в библиотеку новых и полезных для человека функциональностей.

Интеллект – способность субъекта природы к познанию и созиданию для формулирования и решения проблем, связанных с повышением качества жизненного цикла при ограниченных ресурсах времени и средств. Здесь можно не упоминать примитивизмы: самообучение и самосовершенствование, которые направлены только на себя любимого.

На рис. 2 представлены графики развития двух составляющих интеллекта во времени, по годам обучения в университете. Наиболее креативными являются студенты третьего курса, когда их творческий потенциал уже подкреплен полученной технологической и математической культурой, которые способны материализовать еще не покинувшие его фантастические идеи. Что касается циклов активности человека, то можно предположить наличие синус-косинусоидальной (познание-созидание) зависимости, возрастающей по логарифмическому закону от возраста: $f(t) = \sin T + \cos T + \log_2 T$.

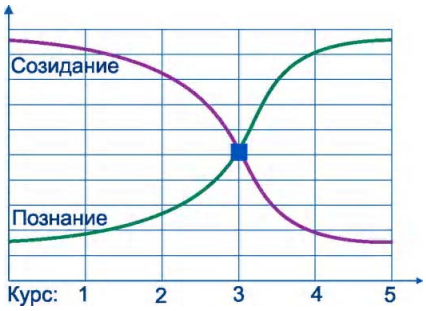


Рис. 2. Составляющие интеллекта в развитии

Как человек берет лучшие функциональности у субъектов природы для расширения своих возможностей, так и компьютер имеет целью совершенствовать себя (окружающий мир), заимствуя или подсматривая у природы интересные решения: кремний – для выполнения элементарных логических функций; математику – для формализации и решения проблем; естественные языки – для создания спецификации и интерфейса проекта; функциональность человеческого мозга, формально выделенную в векторно-логический анализ – для организации структур данных, поиска, распознавания и принятия решений.

Интересно уже сейчас ответить на вопрос, каким будет кибернетическое пространство, компьютер и сервисы через 10 лет, или представить новые перспективные направления развития компьютера и Internet как единой информационно-компьютерной экосистемы, ориентированной на повышение качества жизни путем ее непрерывного использования для поиска, распознавания и принятия решений. При этом продукция от Intel и Microsoft рассматривается как всем доступный инструмент, применяемый для построения стандартизованных и специализированных под каждого пользователя мозгоподобных и дружественных для каждого человека планеты персональных серверов. Сколько людей – столько и компьютеров, *индивидуально* настроенных на каждого конкретного человека. Рыночная привлекательность – все население и все компьютеры планеты. Опасности – выход из-под контроля человека саморазвивающейся информационно-компьютерной экосистемы планеты.

Что можно сделать? 1) Убрать из компьютера тяжеловесную арифметику для повышения быстродействия мозгоподобных (ассоциативно-логических) задач на несколько порядков. 2) Создать стандартизованные структуры данных – иерархия таблиц – для ИКЭС. 3) Разработать параллельный логический ассоциативный мультипроцессор без использования арифметических операций. 4) Создать простые, эффективные метрики и критерии оценивания получаемых решений

в векторно-логическом информационном пространстве. 5) Разработать процесс-модели, реализующие рыночно привлекательные функциональности, ориентированные на поиск, распознавание и принятие решений в ассоциативном векторно-логическом пространстве. 6) Создать инфраструктуру, ориентированную на автоматическое генерирование процесс-моделей поиска, распознавания и принятия решений в кибернетическом пространстве планеты.

3. Эволюция Cyber Space и Internet

Для создания схемы, реализующей полезную функциональность, следует генерировать примитивы $P = \{P_1, P_2, ..., P_i, ..., P_n\}$ самого нижнего уровня. Для этого необходимо создавать фильтры $F = \{F_1, F_2, ..., F_j, ..., F_m\}$, формирующие таблицы примитивных отношений, взятых из информационного пространства планеты (рис. 3). Имея стандартизованные структуры данных для отдельных порталов и браузеров, доставляющих новые сервисы с более высоким быстродействием, следует ожидать постепенного качественного улучшения всех компонентов Cyber Space. Конечная цель такого взаимного и положительного влияния элементов инфраструктуры кибернетического пространства – выработка единых стандартов по интерфейсам и его превращение в саморазвивающуюся интеллектуальную информационно-компьютерную экосистему. Существенное значение будут иметь первичные фильтры или преобразователи для создания новых стандартизованных примитивов, создающих технологичную инфраструктуру для скоростного драйва по Cyber Space с использованием специализированного неарифметического двигателя (I-Computer). Со временем аморфная или «мусорная» часть Internet будет уменьшаться, а стандартизованная инфраструктура – увеличиваться. К 2020 году информационное пространство планеты должно принять цивилизованные форматы структур данных со стандартизованными интерфейсами, подобно тому, как это происходило с развитием планетарной инфраструктуры транспортных сообщений с терминалами, отелями, заправками, ориентированными на сервисы, удовлетворяющие любые запросы пользователя.

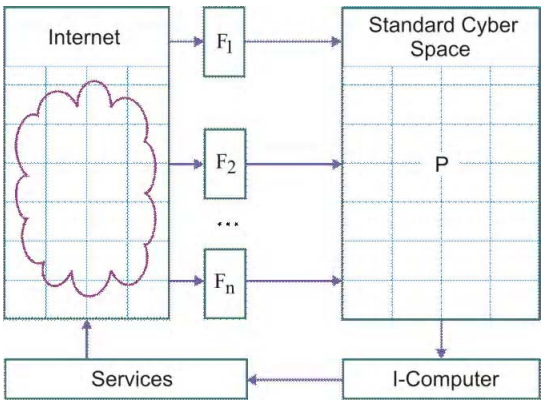


Рис. 3. Эволюция Cyber Space и Internet

В настоящее время отсутствуют стандарты формирования и хранения структур однотипных данных на сайтах и порталах Internet, хотя в сетях базы данных имеют высокий уровень локальной стандартизации. Аморфность глобальной сети затрудняет понимание информации поисковыми системами для распознавания и принятия правильных решений. Трудно ожидать, что информационное пространство в одночасье перейдет на рельсы принятых всеми форматов и интерфейсов. Единственным выходом может служить эволюционирование структур данных. Для этого необходимо разрабатывать преобразователи (фильтры) форматов данных. Наличие первичных фильтров позволяет автоматизировать времязатратные процессы создания библиотек базовых примитивов. Имея спецификацию (рис. 4), представленную после обработки вербального описания в форме вектора входных и выходных переменных, нетрудно записать стратегию создания новой функциональности как задачу поиска покрытия библиотечными элементами обобщенного вектора <X,Y>. Общее решение задачи похоже на синтез автоматной модели, определяющей взаимодействие компонентов во времени и в пространстве. Однако многообразие примитивов, заранее не заданных, исключает такую возможность, что означает – необходим переход из строгого детерминизма цифровых автоматов в область эволюционных, но детерминированных решений.

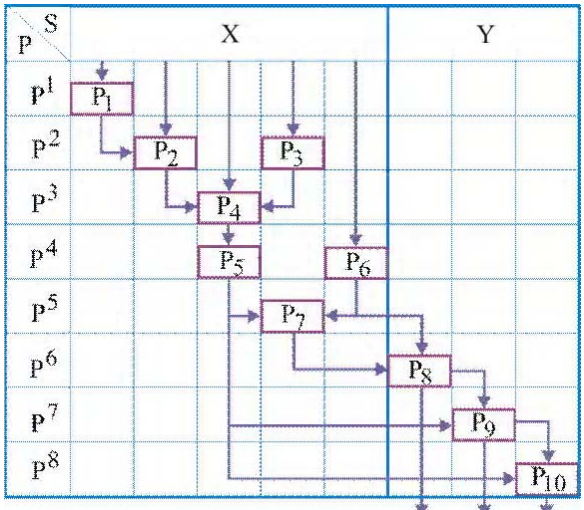


Рис. 4. Синтез покрытия спецификации примитивами

Интеллект f формулируется в виде двух функций (g-созидание и h-повторение), где C, R – процессы созидания и повторения; N, L – примитивы (новый и существующие в библиотеках):

$$I = f(C,R) = C \oplus R;\ 1) C = g(R,N) = R \oplus N;\ 2) R = h(C,L) = C \oplus L$$

1) Генерирование оригинальной функциональности в форме вектора спецификации новых полезных для человека или компьютера сервисов. 2) Синтез функциональной структуры путем покрытия существенных переменных вектора спецификации минимальным множеством примитивов из доступных библиотек планеты для формирования выходного вектора полезных свойств. Повторение двух упомянутых выше

пунктов для создания новой примитивной функциональности, необходимой при решении задачи покрытия. Здесь прослеживаются две спирали развития субъекта киберпространства. Одна идет вверх, по пути создания новых структурных спецификаций. Другая – вниз, по пути создания новых примитивов, обозначающих появление оригинальных технологий.

4. Метрика киберпространства

Пусть имеется $n \neq 0$ конечное число точек в пространстве, составляющее цикл, где каждая из них задана двоичным вектором, длины k:

$$A = (A_1, A_2, ..., A_i, ..., A_n) = \{(a_{11}, a_{12}, ..., a_{1j}, ..., a_{1k}),$$
$$(a_{21}, a_{22}, ..., a_{2j}, ..., a_{2k}), ..., ..., (a_{i1}, a_{i2}, ..., a_{ij}, ..., a_{ik}), ...,$$
$$(a_{n1}, a_{n2}, ..., a_{nj}, ..., a_{nk})\}, a_{ij} = \{0,1\}.$$

Расстояние между двумя точками определяется в виде: $d_i = d_i(A_i, A_{i+1}) = a_{i,j} \overset{k}{\underset{j=1}{\oplus}} a_{i+1,j}$.

Метрика β кибернетического или векторного логического двоичного пространства определяется нулевой хог-суммой расстояний d_i между ненулевым и конечным числом точек, замкнутых в цикл:

$$\beta = \overset{n}{\underset{i=1}{\oplus}} d_i = 0. \qquad (1)$$

Иначе: метрика β киберпространства есть равная нулю хог-сумма расстояний между конечным числом точек, образующих цикл. Сумма k-мерных двоичных векторов, задающих координаты точек циклической фигуры, равна нулю. Данное определение метрики ставит во главу угла не элементы множества, а отношения, что позволяет сократить систему аксиом с трех до одной и распространить ее действие на сколь угодно сложные конструкции n-мерного пространства. Классическое задание метрики для определения взаимодействия одной, двух и трех точек в векторном логическом пространстве, является частным случаем β-метрики при $i = 1,2,3$ соответственно:

$$M = \begin{cases} d_1 = 0 \leftrightarrow a = b; \\ d_1 \oplus d_2 = 0 \leftrightarrow d(a,b) = d(b,a); \\ d_1 \oplus d_2 \oplus d_3 = 0 \leftrightarrow d(a,b) \oplus d(b,c) = d(a,c). \end{cases}$$

Метрика β векторного логического многозначного пространства, где каждая координата определена в алфавите, составляющем булеан $a_{ij} = \{\alpha_1, \alpha_2, ..., \alpha_r, ..., \alpha_m\}$, есть равная пустому множеству симметрическая разность расстояний между конечным числом точек, образующих цикл:

$$\beta = \overset{n}{\underset{i=1}{\Delta}} d_i = \varnothing. \qquad (2)$$

Равенство пустому множеству симметрической разности теоретико-множественного взаимодействия подчеркивает равнозначность компонентов (расстояний), участвующих в формировании уравнения, где единственная операция, используемая в аналитической, четырехзначной модели, определяется Δ-таблицей:

Δ	0	1	x	$\varnothing$
0	$\varnothing$	x	1	0
1	x	$\varnothing$	0	1
x	1	0	$\varnothing$	x
$\varnothing$	0	1	x	$\varnothing$

$\cap$	0	1	x	$\varnothing$
0	0	$\varnothing$	0	$\varnothing$
1	$\varnothing$	1	1	$\varnothing$
x	0	1	x	$\varnothing$
$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$

$\cup$	0	1	x	$\varnothing$
0	0	x	x	0
1	x	1	x	1
x	x	x	x	x
$\varnothing$	0	1	x	$\varnothing$

a	0	1	x	$\varnothing$
$\tilde{a}$	1	0	$\varnothing$	x

(3)

Здесь также приведены таблицы истинности для других базовых теоретико-множественных координатных операций, далее используемых по тексту. Число примитивных символов, образующих замкнутый относительно теоретико-множественных координатных операций алфавит, может быть увеличено. При этом мощность алфавита (булеана) определяется выражением $m = 2^p$, где p – число примитивов.

На основе введенной метрики киберпространства можно формировать квазиоптимальное информационное пространство планеты. Оно не привязано к топологии поверхности и в этом смысле является виртуальным. Но в части покрытия поверхности и пространства телекоммуникациями физическая основа Cyber Space должна быть мажорированной в целях обеспечения отказоустойчивости и надежности функционирования каналов связи при возникновении катаклизмов. Информационная структура Cyber Space должна быть иерархической и замкнутой, как глобально, так и локально, на любом уровне иерархии. Элементарная ячейка структуры пространства должна быть треугольной. Это обеспечит существенное уменьшение информационных объемов Cyber Space, в пределе – на треть. Что означает повышение производительности всех приемо-передатчиков и хранилищ контента планеты на 33%? Данное утверждение связано с аксиомой транзитивного замыкания конечного числа (1,2,3,4,…,n) точек в информационном векторно-логическом пространстве:

1) $d_1 = 0$;
2) $d_1 \oplus d_2 = 0$;
3) $d_1 \oplus d_2 \oplus d_3 = 0$;
4) $d_1 \oplus d_2 \oplus d_3 \oplus d_4 = 0$;
5) $d_1 \oplus d_2 \oplus ... \oplus d_i \oplus ... \oplus d_n = 0$.

Рассматривая все возможные варианты транзитивного замыкания с позиции минимизации условий для восстановления информации, очевидным представляется факт восстановления третьей стороны треугольника по двум известным. В этом случае достаточ-

но, например, передать по каналу связи две стороны, чтобы восстановить третью. Сокращение передаваемых объемов в данном случае равно 33%. Во всех других случаях уменьшение объемов информации будет меньшим. Например, для замкнутого четырехугольника необходимо передать три расстояния или стороны, чтобы восстановить четвертую, выигрыш – 25%. Таким образом, плоскостная интерпретация Cyber Space будет иметь вид, представленный на рис. 5.

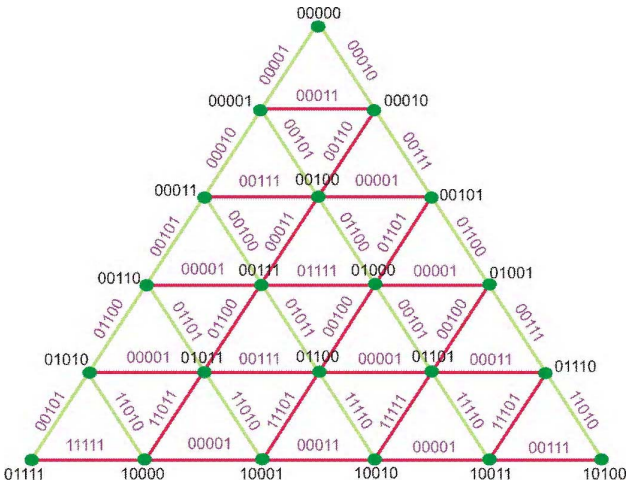


Рис. 5. Triangle Cyber Space

Для того чтобы восстановить всю упомянутую структуру, необходимо иметь (передать) уже не 2/3 объема информации, которая соответствует каждым двум сторонам из трех всех треугольников, представленных зеленым (более светлым) цветом. Для данной структуры достаточно передать стороны, заданные зеленым цветом, что составляет 44% от объема кодов-расстояний. Остальные 56% компонентов треугольников могут быть определены с помощью аксиомы: $d_1 \oplus d_2 \oplus d_3 = 0 \rightarrow d_3 = d_1 \oplus d_2$. В общем случае функциональная зависимость отношения восстанавливаемых сторон замкнутого в треугольник пространства к общему числу n слоев треугольной структуры определяется выражением:

$$\eta = \frac{\sum (i+1)}{\sum (3 \times i)}\Big|_{i=\overline{1,n}} = \frac{1}{3}\left(\frac{2}{n+1}+1\right) =$$

$$= \frac{1}{3} \cdot \frac{2+n+1}{n+1} = \frac{1}{3} \cdot \frac{n+3}{n+1} = \frac{n+3}{3(n+1)}.$$

График отношения количества передаваемой информации к ее полному восстанавливаемому объему расстояний в треугольном кибернетическом пространстве представлен на рис. 6. Предельное значение графика стремится к 33%. Это означает – для больших информационных объемов треугольного пространства достаточно передать третью часть всех кодов-расстояний, чтобы затем восстановить полное пространство кодов.

Треугольная метрика пространства есть самая экономичная, поскольку она создает самые короткие расстояния и пути между объектами, благодаря наличию транзитивного замыкания. Соседство объекта с шестью точками делает метрику более предпочтительной по сравнению с «Манхэттеном» в части выбора путей для достижения объекта в пространстве. Оптимальная структура мультипроцессорной системы должна быть составлена из треугольников. Здесь оптимальность определяется соотношением длины пути между любыми двумя точками и общим числом сторон, формирующих треугольное пространство. Длина пути здесь всегда будет не хуже, чем в «Манхэттене», но общее число связей больше на одну диагональ в каждом четырехугольнике. Это дает возможность существования прямых контактов для 6 соседей, что существенно для мультипроцессора. «Манхэттен» с диагональными связями обладает той же длиной пути, что и треугольное пространство, но имея при этом на одну связь больше. Здесь задействована вторая диагональ четырехугольника, что обеспечивает соседство с восемью вершинами пространства. Поскольку треугольник – самая примитивная фигура, формирующая плоскость, то вес одной стороны, которую не следует хранить для идентификации пространства, будет максимальным в треугольнике. Далее собранные в систему треугольники дают уже 66% структурных компонентов, которые можно не описывать при формировании пространства, но которые при желании можно восстановить. Все другие примитивные плоскостные фигуры дают меньший выигрыш при формировании ими Cyber Space.

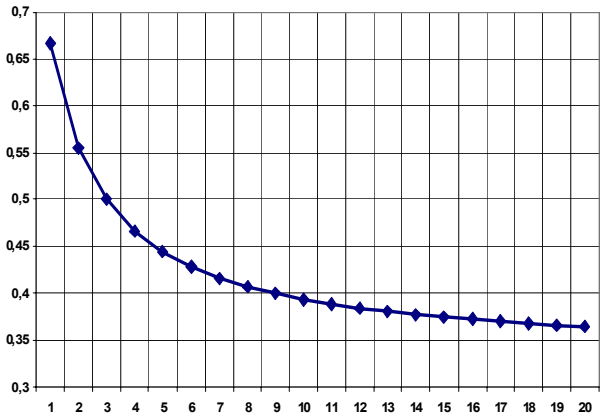


Рис. 6. Функция числа расстояний для восстановления пространства

5. Типы взаимодействия объектов и интегральная оценка

Кодовое расстояние по Хэммингу дает возможность численно оценить меру равенства объектов или абсолютную оценку числа несовпадений в разрядах двух векторов. Недостатками такого измерения являются: 1) Отсутствие относительной оценки взаимодействия объектов – одно несовпадение на ста разрядах или на двух – две большие разницы. 2) Скрытость в скалярной оценке числа координат, по которым произошло несовпадение, представляется деструктивной для анализа результата. 3) Неспособность адекватно и однозначно оценивать все типы взаимодействия двух

объектов, представленных на рис. 7. Здесь существует скалярная неразличимость оценивания первого и третьего, второго и четвертого вариантов отношений между m и A.

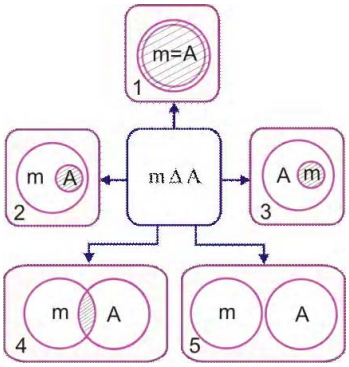


Рис. 7. Типы взаимодействия двух объектов

Упомянутые выше недостатки определяют необходимость создания универсальной неарифметической оценки взаимодействия двух объектов в киберпространстве, которая должна иметь одинаковый с оцениваемыми объектами формат, выраженную структуру взаимодействия переменных и распознавать любые типы теоретико-множественного или векторного взаимодействия объектов. Важно также создать аппаратную поддержку быстрого путешествия по киберпространству в целях поиска информации, распознавания образов и принятия решений [4] на основе векторных неарифметических операций. Оценка бинарного отношения или решения задачи определяется векторно-логическим критерием качества взаимодействия запроса (вектора m) с системой ассоциативных векторов (ассоциаторов), в результате которого генерируется конструктивный ответ в виде одного или нескольких ассоциаторов, а также, пока еще, численной характеристики степени принадлежности (функции качества) входного вектора m к найденному решению: $\mu(m \in A)$. Входной вектор $m = (m_1, m_2, ..., m_i, ..., m_q)$, $m_i \in \{0,1,x\}$ и матрица A_i ассоциаторов (векторов) $A_{ijr}(\in A_{ij} \in A_i \in A) = \{0,1,x\}$ имеют одинаковую размерность, равную q. Далее степень принадлежности m-вектора к вектору A будет обозначаться как $\mu(m \in A)$.

Существует 5 типов теоретико-множественного (логического) Δ-взаимодействия двух векторов $m \cap A$, определенных ранее на рис. 7. Они формируют все примитивные варианты реакции некоторой системы на входной вектор-запрос. В технической диагностике (Design & Test) – указанная последовательность действий изоморфна маршруту: поиск дефектов, их распознавание, принятие решения на восстановление работоспособности. Все три стадии технологического маршрута нуждаются в метрике оценивания решений для выбора оптимального варианта.

Определение. Интегральная теоретико-множественная метрика для оценивания качества запроса есть функция взаимодействия многозначных векторов $m\Delta A$, которая определяется средней суммой трех нормированных параметров: кодовое расстояние $d(m, A)$, функция принадлежности $\mu(m \in A)$ и функция принадлежности $\mu(A \in m)$:

$$
\begin{aligned}
&Q = \frac{1}{3}[d(m,A) + \mu(m \in A) + \mu(A \in m)],\\
&d(m,A) = \frac{1}{n}[n - \mathrm{card}\,(m_i \overset{k}{\underset{i=1}{\cap}} A_i = \varnothing)];\\
&\mu(m \in A) = 2^{c-a};\\
&\mu(A \in m) = 2^{c-b};\\
&a = \mathrm{card}\,(\overset{k}{\underset{i=1}{\cup}} A_i = x);\\
&b = \mathrm{card}\,(\overset{k}{\underset{i=1}{\cup}} m_i = x);\\
&c = \mathrm{card}\,(m_i \overset{k}{\underset{i=1}{\cap}} A_i = \ldots);
\end{aligned}
\qquad (4)
$$

Пояснения. Пересечение (объединение) векторов – есть векторная логическая операция, основанная на соответствующих координатных теоретико-множественных операциях. Операции координатного пересечения и объединения определены в алфавите Кантора $\{0,1,x = \{0,1\},\varnothing\}$. Нормирование параметров позволяет оценить уровень взаимодействия векторов в интервале [0,1]. Если зафиксировано предельное максимальное значение каждого параметра, равное 1, то векторы равны между собой. Минимальная оценка, $Q = 0$, фиксируется в случае полного несовпадения векторов по всем n координатам. Следует заметить, если пересечение двух векторов равно пустому множеству, то степень двойки от символа «пусто» равна нулю: $2^{\mathrm{card}(m \cap A) = \varnothing} = 2^{\varnothing} = 0$. Это действительно означает, что количество общих точек при пересечении двух пространств, формируемых двумя векторами, равно нулю.

Цель введения векторно-логического критерия качества решения заключается в существенном повышении быстродействия при подсчете оценки Q взаимодействия компонентов m и A при анализе ассоциативных структур данных путем использования только векторных логических операций. Арифметический критерий (4) без усреднения функций принадлежности и кодового расстояния можно трансформировать к виду:

$$
\begin{aligned}
&Q = d(m,A) + \mu(m \in A) + \mu(A \in m),\\
&d(m,A) = \mathrm{card}\,(m_i \overset{k}{\underset{i=1}{\oplus}} A_i = U);\\
&\mu(m \in A) = \mathrm{card}\,(A_i = U) - \mathrm{card}(m_i \overset{k}{\underset{i=1}{\wedge}} A_i = U);\\
&\mu(A \in m) = \mathrm{card}(m_i = U) - \mathrm{card}(m_i \overset{k}{\underset{i=1}{\wedge}} A_i = U);\\
&U = \begin{cases} 1 \leftarrow \{m_i, A_i\} = \{0,1\}; \\ x \leftarrow \{m_i, A_i\} = \{0,1,x\}. \end{cases}
\end{aligned}
\qquad (5)
$$

Представленные здесь векторные логические операции изоморфны $(\wedge,\vee,\oplus,\neg) \approx (\cap,\cup,\Delta,\sim)$ теоретико-множественным. При этом теоретико-множественные координатные операции, соответствующие данным логическим, были определены ранее на многозначном алфавите Кантора в выражении (5). Первый компонент, составляющий критерий, формирует степень несовпадения n-мерных векторов – кодовое расстояние, путем выполнения операции xor, второй и третий определяют степень непринадлежности результата конъюнкции к числу единиц каждого из двух взаимодействующих векторов. Понятия принадлежности и непринадлежности являются взаимодополняющими, но в данном случае технологичнее вычислять непринадлежность. Следовательно, необходимый критерий качества равен нулю, когда два вектора равны между собой. Оценка качества взаимодействия двух двоичных векторов убывает по мере возрастания критерия от нуля до единицы. Для того чтобы окончательно исключить арифметические операции при подсчете векторного критерия качества, необходимо объединить три оценки в одну:

$$
\begin{aligned}
Q &= d(m,A) \vee \underline{\mu(m \in A)} \vee \mu(\underline{A \in m}) = \\
&= m \oplus A \vee A \wedge m \wedge \underline{A} \vee m \wedge \underline{m} \wedge A = \\
&= m \oplus A \vee [A \wedge (\overline{m} \vee \underline{A})] \vee [m \wedge \underline{(\overline{m} \vee \overline{A})}] = \\
&= m \oplus A \vee [\underline{A\overline{m}} \vee A\underline{A} \vee m\overline{m} \vee \underline{mA}] = \\
&= (A\overline{m} \vee \underline{mA}) \vee [A\overline{m} \vee \underline{A\overline{A}} \vee m\overline{m} \vee \underline{mA}] = \\
&= A\overline{m} \vee m\overline{A} \vee A\overline{m} \vee A\overline{A} \vee m\overline{m} \vee m\overline{A} = \\
&= m \oplus A.
\end{aligned}
$$

Процедура вычисления векторного критерия качества зависит от значности алфавита:

$$
Q = \begin{cases} m \oplus A \leftarrow \{m_i, A_i\} = \{0,1\}; \\ m \Delta A \leftarrow \{m_i, A_i\} = \{0,1,x\}. \end{cases} \qquad (6)
$$

В случае, когда алфавит описания координат имеет три значения, вычисление симметрической разности осуществляется в соответствии с Δ - операцией, определенной ранее. Критерий качества логически объединяет кодовое расстояние и функции принадлежности. Алфавит, в свою очередь, разделяет данные оценки на два случая. Если он двоичный, то функции принадлежности равны нулю (существует различие, хотя бы по одной координате) или единице (векторы идентичны). Для многозначного алфавита существование хотя бы одного пустого координатного пересечения также уничтожает функции принадлежности. В противном случае их логическое объединение по or-функции порождает симметрическую разность, которая в двоичном алфавите носит название xor-операции. Таким образом, кодовое расстояние по Хэммингу равно функциям принадлежности и равно метрике векторно-логического пространства:

$$
Q = m \oplus A = \begin{cases} d(m,A) = m \oplus A \leftarrow \{m_i, A_i\} = \{0,1\}; \\ \mu(m \in A) \vee \mu(A \in m) = m \Delta A \leftarrow \{m_i, A_i\} = \{\alpha_1, \alpha_2, \ldots \alpha_i, \ldots \alpha_k\}. \end{cases}
$$

В качестве примера далее представлена оценка бинарного взаимодействия четырех пар многозначных векторов:

$$
\begin{aligned}
Q_1 &= m_1(1xxxx010) \oplus A_1(10xx10xx) = (\varnothing 1 \varnothing 0 \varnothing 01); \\
Q_2 &= m_2(1xxxx001) \oplus A_2(10xx1010) = (\varnothing 1 \varnothing 0 \varnothing UU); \\
Q_3 &= m_3(10000010) \oplus A_3(10000010) = (00000000); \\
Q_4 &= m_4(11111001) \oplus A_4(00011001) = (11100000).
\end{aligned}
$$

В первом случае кодовое расстояние теряет свой смысл, поскольку два объекта имеют общее пространство

$$
m_1(1xxxx010) \cap A_1(10xx10xx) = (10xx1010) .
$$

Здесь необходимо определить взаимодействие многозначных векторов в виде функций принадлежности с помощью операции симметрической разности. Это означает, что одна метрика работает в различных по алфавиту пространствах для оценивания бинарных отношений векторно-логических объектов. Во втором случае число координат, равное универсальной единице (U), формирует кодовое расстояние бинарного взаимодействия. Равно как и универсальный ноль ($\varnothing$) идентифицирует координатные равенства векторной логической оценки. Третья оценка фиксирует равенство объектов, заданных в двоичном алфавите описания координат. Здесь, говоря языком скалярных отношений, кодовое расстояние равно нулю, а функции принадлежности имеют максимальное единичное значение. Четвертая оценка есть расстояние между двоичными векторами, равное трем координатам.

Критерий качества Q однозначно определяет три формы взаимодействия двух любых объектов в n-мерном векторном логическом пространстве: расстояние и две функции принадлежности, равные расстоянию. Увеличение числа нулей повышает критерий качества, а наличие единиц обусловливает ухудшение качества взаимодействия по соответствующим булевым переменным. Критерий качества $Q = m \oplus A$ согласуется с метрикой оценивания расстояния или взаимодействия объектов в векторно-логическом пространстве, а также имеет тривиальную вычислительную процедуру для оценивания решений, связанных с анализом и синтезом информационных объектов. В самом деле, векторное логическое пространство не должно иметь метрического расстояния и численных критериев качества, включающих арифметические операции на скалярных величинах. Для сравнения оценок необходимо определять мощность единиц в каждом векторе без выполнения операций суммирования. Это можно сделать с помощью регистра [14], который позволяет за один такт выполнить сдвиг влево и уплотнить все единичные координаты n-разрядного двоичного вектора.

Процесс-модель поиска оценки лучшего решения из более чем двух альтернатив представлена на рис.8 и имеет следующие пункты. 1) Первоначально в вектор-результат Q, в котором будет сохранено лучшее решение, заносятся единичные значения во все координаты (худшее решение) и одновременно осуществляется операция slc сдвига влево с уплотнением единиц текущего вектора Q_i. 2) Выполняется сравнение двух векторов: Q и очередной оценки Q_i из списка решений. 3) Реализуется векторная операция and $(Q \wedge Q_i)$, результат которой сравнивается с содержимым вектора Q, что дает возможность изменить его, если вектор Q_i имеет меньшее число единичных значений. 4) Процедура поиска оценки лучшего решения повторяется n раз:

$$Q = Q\overline{(\vee((Q \wedge Q_i) \oplus Q))} \vee Q_i(\vee((Q \wedge Q_i) \oplus Q));$$
$$Y = \vee((Q \wedge Q_i) \oplus Q);$$
$$Q = Q\overline{Y} \vee Q_i Y.$$

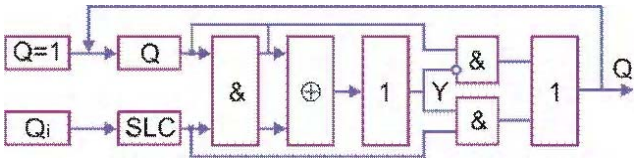


Рис. 8. Процесс-модель выбора решения

Векторные логические критерии качества взаимодействия ассоциативных наборов позволяют получать оценку поиска, распознавания и принятия решения с высоким быстродействием логических параллельных операций на основе специализированного процессора, что особенно существенно для критических систем реального времени.

6. Двигатель для киберпространства

Для скоростного путешествия по киберпространству (поиск объектов и оценка их взаимодействия) нужен простой и быстродействующий мультиматричный процессор (ММП), где каждая команда (and, or, xor, slc) обрабатывает параллельно и предельно быстро только одну бинарную операцию на матрицах (двумерные массивы данных). Количество командно-ориентированных матриц-примитивов создает систему – гетерогенный мультиматричный процессор бинарных операций с буфером M (рис. 9). Мультиматричный модуль процессора включает 4 блока памяти со встроенными на них операциями (A – and, B – xor, C – slc – shift left crowding, D – or) и буферную память M. Модуль ориентирован на параллельное выполнение в данном случае одной из четырех инструкций (ISA – Instruction Set Architecture), оперирующих матрицами двоичных данных одинаковой размерности: $M = M\{and, or, xor, slc\}\{A, B, C, D\}$ с занесением результата в буфер M.

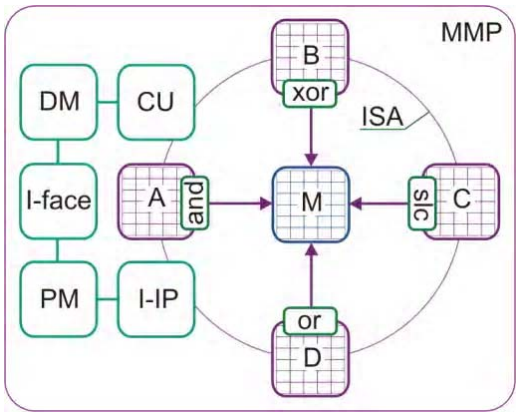


Рис. 9. Мультиматричный процессор бинарных операций

Особенность ММП в том, что не ячейка матрицы имеет систему команд из четырех операций, а каждая команда имеет собственную матрицу ячеек в качестве данных для параллельной обработки, что существенно упрощает структуру управления и устройства в целом. Вся сложность ММП перенесена на структуры данных, где память матрицы имеет одну аппаратно-реализованную встроенную команду, что позволяет иметь примитивную систему управления параллельными вычислительными процессами (SIMD – Single Instruction Multiple Data), последовательностную по своей сути, а значит, нет необходимости создавать сверхсложные компиляторы, ориентированные на распараллеливание вычислительных процессов. Здесь каждый матричный процессор выполняет одну операцию, встроенную в запоминающие элементы матрицы. Но возникают ситуации, когда матричный уровень (M-level) задания данных избыточен для выполнения, например операций над булевыми (B-level) или регистровыми (R-level) переменными. Для такого случая необходимо иметь иерархию по уровням представления данных. Стандартные блоки ММП: памяти данных DM и программ PM, управления CU, интерфейс I-face и сервисного обслуживания I-IP, а также мультиматричный модуль процессора, включающий 4 блока памяти со встроенными в них операциями (A – and, B – xor, C – or, D – slc – shift left crowding) и буферную память M.

Практическая значимость мультиматричного процессора:

1. Векторная и матричная оценки степени сходства прообразов или взаимодействия объектов любой природы вычисляются параллельно, путем обработки векторов или матриц, использующих операцию симметрической разности, определяемую за один временной такт благодаря применению мультиматричного процессора.

2. Наглядность матричной оценки в фрагментах изображения, где существуют различия, дает возможность корректировать не только анализируемый объект, но и эталонные примитивы на стадии создания библиотеки образов.

3. Матричный двоичный образ является одним из перспективных вариантов применения технологии распознавания, поскольку все объекты в кибернетическом пространстве, по сути, есть цифровые образы, к которым применима универсальная модель векторно-логического процесса распознавания, предложенная выше.

4. Стандартизация моделей образов. Разбиение образа на сегменты позволяет масштабировать изображения любой размерности для их приведения к матрице одной размерности, имеющей одинаковое число координат, которые идентифицируются двоичными или многозначными значениями, составляющими сигнатуру изображения. Это дает возможность стандартизировать модели входных образов, эталонов и критериев под эгидой одинакового формата данных. Имея два любых из упомянутых компонентов, можно легко восстановить третий. Например, взаимодействие матриц одной размерности образа, эталона и критерия качества взаимодействия может быть представлено тремя полезными равенствами:

$$m \oplus A \oplus Q = 0 \rightarrow \begin{cases} Q = m \oplus A; \\ m = A \oplus Q; \\ A = m \oplus Q \end{cases}$$

для решения сотен практически полезных задач, включая: принятие решений, распознавание образов, тестирование и диагностику процессов и явлений. Области применения векторно-логической или матричной технологии анализа процессов или явлений: 1) распознавание текстов в регистрационных картах пересечения границы; 2) идентификация личности по фотографиям, близким к стандартным снимкам, для получения визовых документов; 3) поиск аналогов в Интернете по заданным образам; 4) сортировка изображений в базе данных по классам и признакам; 5) дактилоскопия и создание классифицированной библиотеки спецслужб; 6) распознавание и классификация программных вирусов; 7) распознавание почерков и идентификация личности по существенным признакам написания букв; 8) идентификация наземных целей и движущихся объектов (самолеты, корабли, машины); 9) синтез или корректировка образов по характерным существующим признакам; 10) управление робототехническими комплексами; 11) распознавание образов обоняния, вкуса, звуковых, тепловых и радиочастотных излучений; 12) распознавание лингвистических конструкций и примитивов, а также их оценивание при сравнении с эталонами; 13) динамическая визуализация эффектов последовательного и мягкого трансформирования одного образа в любой другой.

7. Инфраструктура программного продукта

Проблема создания эффективной инфраструктуры кибернетического пространства (Cyber Space), а также саморазвивающейся информационно-компьютерной экосистемы (ИКЭС) планеты особенно важна для глобальных компаний, таких как Лаборатория Касперского, Google, Microsoft, Apple, Intel. Экосистема – совокупность популяций, взаимодействующих между собой и окружающей их средой неопределённо долгое время, имеющая сходство относительно протекающих в них энергетических (информационных) процессов.

Кибернетическое пространство как объект природы также подвержено влиянию деструктивных компонентов, влияющих на работоспособность субъектов, которыми являются компьютеры, системы и сети. Поэтому сейчас и в будущем важной проблемой остается стандартизация пространства и специализация всех взаимодействующих субъектов, включая негативные, как неотъемлемую часть экосистемы. Данная акция есть постоянно действующая во времени, цель которой – не отставать, но на один шаг опережать появление новых вредоносных компонентов.

Для этого необходимо разработать инфраструктуру сервисного обслуживания программного продукта для защиты от вредоносных программ, которая включает: 1) Наблюдение за состоянием компьютерных систем и сетей в процессе функционирования и тестирования штатных функциональных блоков на основе использования стандартов ассерционного анализа.

2) Тестирование функциональных модулей путем подачи проверяющих наборов от различных тестовых генераторов, ориентированных на проверку вредоносов или исправного поведения.

3) Диагностирование отказов и вредоносов путем анализа информации, полученной на стадии тестирования и использования специальных методов встроенного поиска ВК на основе ассерционной избыточности, ориентированной на обнаружение вредоносного кода, что позволит идентифицировать и устранять деструктивы программных продуктов без использования внешних средств. Таким образом, можно будет обходиться без сложных внешних программ моделирования, тестирования и диагностирования путем прививки каждого программного изделия тестопригодной интеллектуальной избыточностью кода на стадии его создания. При этом следует использовать предикат узнавания буквы a: $x^a = x \oplus a$, который оперирует не только булевыми, но регистровыми и матричными переменными, что делает его практически значимым в формальной записи уравнений диагноза или распознавания:

$$x^a \approx x \oplus a = 0 \vee \min_i Q_i \rightarrow x \oplus a \oplus Q = 0.$$

На основе предиката узнавания a-образа любой сложности, природы и формы можно создавать достаточно компактные уравнения предикатов, формирующие интеллектуальные решения в области распознавания образов, принятий решений, тестирования знаний и технических объектов, диагностирования (узнавания) вирусных компонентов в программном коде.

4) Восстановление работоспособности функциональных модулей после фиксации отрицательного результата тестирования и определения места и вида вредоноса при выполнении фазы диагностирования внутренними средствами, создающими инфраструктуру функциональности.

5) Измерение сигнатуры, основных характеристик и параметров функционирования изделия на основе встроенных средств, позволяющих производить временные и функциональные измерения для идентификации состояния программного продукта.

6) Надежность и отказоустойчивость функционирования изделия в процессе эксплуатации, которая достигается диверсификацией функциональных блоков, их дублированием и восстановлением работоспособности в реальном масштабе времени, а также встроенными средствами сервисного обслуживания, работающими в реальном масштабе времени.

В связи с этим предложенные компоненты инфраструктуры кибернетического пространства, метрика его измерения и процесс-модели анализа и синтеза субъектов дают возможность создавать эффективные решения компьютерных устройств, ориентированных на быстрый поиск, распознавание, диагностирование не только позитивных, но и негативных субъектов. Конкретно, предложенная инфраструктура может решать задачи: 1) Описание многообразия деструктивных компонентов кибернетического пространства. 2) Формализация процессов взаимодействия триады компонентов <программа, деструктивности, тесты>. 3) Диагностирование и устранение деструктивных компонентов. 4) Создание и эффективное использование базы деструктивных данных. 5) Создание быстродействующих интеллектуальных саморазвивающихся средств сервисного обслуживания и защиты кибернетического пространства.

8. Модель поиска ВК в программных продуктах

Используется уравнение пространства $f(F,T,L,U)=0 \rightarrow F \oplus T \oplus L \oplus U = 0$, которое трансформируется к виду $L=(T \oplus F) \oplus (T \oplus U)$. Диагностирование ВК сводится к сравнению результатов модельного $(T \oplus F)$ и натурного $(T \oplus U)$ экспериментов, которое формирует список функциональных нарушений L, присутствующих в объекте диагностирования. Формула-модель процесса поиска блока F_i с функциональными нарушениями сводится к выбору решения посредством определения хог-взаимодействия между тремя компонентами:

$$L = F_i \leftarrow [(T \oplus F_i) \overset{p}{\underset{i=1}{\oplus}} (T \oplus U_i)] = 0.$$

Аналитическая модель верификации HDL-кода с использованием механизма темпоральных ассерций (дополнительных линий наблюдения) ориентирована на достижение заданной глубины диагностирования и представлена в следующем виде:

$$\begin{aligned} &M = f(F,A,B,S,T,L), && F = (A*B)\times S;\ S = f(T,B);\\ &A = \{A_1,A_2,...,A_i,...,A_n\}; && B = \{B_1,B_2,...,B_i,...,B_n\};\\ &S = \{S_1,S_2,...,S_i,...,S_m\}; && S_i = \{S_{i1},S_{i2},...,S_{ij},...,S_{ip}\};\\ &T = \{T_1,T_2,...,T_i,...,T_k\}; && L = \{L_1,L_2,...,L_i,...,L_n\}. \end{aligned} \quad (7)$$

Здесь $F=(A*B)\times S$ – функциональность, представленная графом (рис. 10) транзакций программных блоков (Code-Flow Transaction Graph – CFTG), где $S=\{S_1,S_2,...,S_i,...,S_m\}$ – вершины или состояния программного продукта при моделировании тестовых сегментов.

Каждое состояние $S_i=\{S_{i1},S_{i2},...,S_{ij},...,S_{ip}\}$ определяется значениями существенных переменных проекта (булевы, регистровые переменные, память). Ориентированные дуги графа представлены конкатенацией программных блоков $B=(B_1,B_2,...,B_i,...,B_n)$ и соответствующих им ассерций $A=\{A_1,A_2,...,A_i,...,A_n\}$. Каждая дуга B_i – группа операторов кода – формирует состояние вершины $S_i=f(T,B_i)$ в зависимости от теста $T=\{T_1,T_2,...,T_i,...,T_k\}$. Вершина может иметь более одной входящей (исходящей) дуги. Множество блоков с функциональными нарушениями представлено списком $L=\{L_1,L_2,...,L_i,...,L_n\}$.

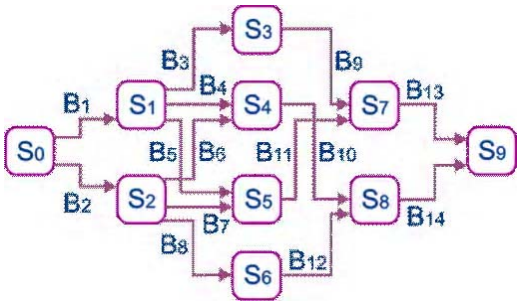


Рис. 10. Пример ABC-графа программы

Модель кода, представленная в форме Assertion Based Coverage Graph (ABC-Graph), отображает не только структуру программного кода, но и тестовые срезы функциональных покрытий, формируемые с помощью программных блоков, входящих в рассматриваемую вершину. Последняя определяет отношение между достигнутым на тесте пространством переменных и потенциально возможным, которое формирует функциональное покрытие как мощность состояния i-вершины графа $Q=\mathrm{card}C_i^r / \mathrm{card}C_i^p$. В совокупности все вершины графа должны составлять полное покрытие пространства состояний переменных программного кода, которое формирует качество теста, равное 1 (100%): $Q=\mathrm{card}\bigcup_{i=1}^{m}C_i^r / \mathrm{card}\bigcup_{i=1}^{m}C_i^p = 1$. Кроме того, механизм ассерций $<A,C>$, существующий на графе, позволяет выполнять мониторинг дуг

(code-coverage) $A = \{A_1, A_2, ..., A_i, ..., A_n\}$ и вершин (functional coverage) $C = \{C_1, C_2, ..., C_i, ..., C_m\}$. Ассерции на дугах отвечают за диагностирование функциональных нарушений в программных блоках. Ассерции на вершинах графа дают дополнительную информацию о качестве тестов (ассерций) в целях их улучшения или дополнения. ABC-граф HDL-кода дает возможность использовать аппарат тестопригодного проектирования для оценки качества программного продукта с позиции минимизации затрат на создание тестов, диагностирование, а также исправление функциональных нарушений. Также граф дает возможность оптимизировать синтез теста путем решения задачи покрытия минимальным множеством активизированных путей всех дуг (вершин). Минимальный тест для приведенного ABC-графа имеет шесть сегментов, которые активизируют все пути:

$$T = S_0S_1S_3S_7S_9 \vee S_0S_1S_4S_8S_9 \vee S_0S_1S_5S_7S_9 \vee$$
$$\vee S_0S_2S_4S_8S_9 \vee S_0S_2S_5S_7S_9 \vee S_0S_2S_6S_8S_9.$$

Тесту можно поставить в соответствие следующую матрицу активизации программных блоков:

B_{ij}	B_1	B_2	B_3	B_4	B_5	B_6	B_7	B_8	B_9	B_{10}	B_{11}	B_{12}	B_{13}	B_{14}
T_1	1	.	1	.	.	.	.	.	1	.	.	.	1	.
T_2	1	.	.	1	.	.	.	.	.	1	.	.	.	1
T_3	1	.	.	.	1	.	.	.	.	.	1	.	1	.
T_4	.	1	.	.	.	1	.	.	.	1	.	.	.	1
T_5	.	1	.	.	.	.	1	.	.	.	1	.	1	.
T_6	.	1	.	.	.	.	.	1	.	.	.	1	.	1

Матрица активизации иллюстрирует факт неразличимости на тесте функциональных нарушений в блоках 3 и 9, 8 и 12, которые составляют два класса эквивалентностей при наличии одной ассерции (монитора) в вершине 9. Для устранения такой неразличимости необходимо создать два дополнительных монитора в вершинах 3 и 6. В результате 3 ассерции $A = (A_3, A_6, A_9)$ дают возможность различить между собой все блоки программного кода. Таким образом, граф позволяет не только синтезировать оптимальный тест, но и определять минимальное число мониторов (ассерций) для поиска неисправных блоков с заданной глубиной диагностирования.

Увеличение числа ассерционных мониторов приводит к модификации таблицы активизации. Иначе, на заданном тесте и механизме ассерций необходимо однозначно решать задачу диагностирования функциональных нарушений программного кода с глубиной до программного блока. При этом число ассерций и тестовых сегментов должно быть минимально допустимым для кодовой идентификации всех блоков: $|T| + |A| \geq \log_2 |B| = \mathrm{card}T + \mathrm{card}A \geq \log_2 \mathrm{card}B$. Первоначально количество мониторов-ассерций равно числу тестовых сегментов. Таблица активизации программных модулей позволяет выполнять идентификацию блоков кода с функциональными нарушениями на обобщенном векторе экспериментальной проверки (ассерционного мониторинга) $V=(V_1, V_2, ..., V_i, ..., V_n), V_i = \{0,1\}, V_i = T_i \oplus B_j, \forall j(B_{ij} = 1)$. Координата вектора $V_i = T_i \oplus B_j = 1$ идентифицирует факт «падения» тест-сегмента на подмножестве активизированных им блоков. В соответствии с вектором V, заданным на таблице активизации с учетом приведенного выше правила вычисления его координат:

B_{ij}	B_1	B_2	B_3	B_4	B_5	B_6	B_7	B_8	B_9	B_{10}	B_{11}	B_{12}	B_{13}	B_{14}	V
T_1	1	.	1	.	.	.	.	.	1	.	.	.	1	.	0
T_2	1	.	.	1	.	.	.	.	.	1	.	.	.	1	1
T_3	1	.	.	.	1	.	.	.	.	.	1	.	1	.	0
T_4	.	1	.	.	.	1	.	.	.	1	.	.	.	1	1
T_5	.	1	.	.	.	.	1	.	.	.	1	.	1	.	0
T_6	.	1	.	.	.	.	.	1	.	.	.	1	.	1	1

можно построить логическую функцию функциональных нарушений программного продукта, которая упрощается на основе использования координат вектора экспериментальной проверки V:

$$B = (\overline{T}_1 \vee B_1 \vee B_3 \vee B_9 \vee B_{13}) \wedge (\overline{T}_2 \vee B_1 \vee B_4 \vee B_{10} \vee B_{14}) \wedge$$
$$\wedge (\overline{T}_3 \vee B_1 \vee B_5 \vee B_{11} \vee B_{13}) \wedge (\overline{T}_4 \vee B_2 \vee B_6 \vee B_{10} \vee B_{14}) \wedge$$
$$\wedge (\overline{T}_5 \vee B_2 \vee B_7 \vee B_{11} \vee B_{13}) \wedge (\overline{T}_6 \vee B_2 \vee B_8 \vee B_{12} \vee B_{14});$$
$$\{V, T\} = (010101) \rightarrow$$
$$B = (0 \vee B_1 \vee B_4 \vee B_{10} \vee B_{14}) \wedge (0 \vee B_2 \vee B_6 \vee B_{10} \vee B_{14}) \wedge$$
$$\wedge (0 \vee B_2 \vee B_8 \vee B_{12} \vee B_{14}) =$$
$$= (B_1 \vee B_4 \vee B_{10} \vee B_{14}) \wedge (B_2 \vee B_6 \vee B_{10} \vee B_{14}) \wedge$$
$$\wedge (B_2 \vee B_8 \vee B_{12} \vee B_{14}) =$$
$$= B_1B_2 \vee B_4B_2 \vee ... \vee B_3B_6B_{12} \vee ... \vee B_{14}.$$

После преобразования КНФ к дизъюнктивной нормальной форме полученные термы содержат все возможные решения в виде покрытия единичных координат вектора экспериментальной проверки одиночными или кратными функциональными нарушениями программных блоков. Выбор лучшего решения связан с определением терма ДНФ минимальной длины. В данном примере оптимальным решением является терм, содержащий один блок $B = B_{14}$, который своим функциональным нарушением покрывает три единицы в векторе экспериментальной проверки $V = (010101)$. Данный факт также очевиден из сравнения двух последних столбцов матрицы активизации B.

Определенный интерес представляет задача однозначной идентификации блока с нарушениями путем анализа матрицы активизации. Для рассматриваемого примера графа и соответствующей ему таблицы необходимо поставить три обобщенных монитора в вершинах 3, 6, 9 ABC-графа. Такое эвристическое решение требует изменения структуры матрицы активизации путем добавления разрядов в некоторых координатах таблицы B (механизме ассерций) и вектора экспериментальной проверки V:

B_{ij}	B_1	B_2	B_3	B_4	B_5	B_6	B_7	B_8	B_9	B_{10}	B_{11}	B_{12}	B_{13}	B_{14}	V	A
T_1	1	.	11	.	.	.	.	.	01	.	.	.	1	.	00	3,9
T_2	1	.	.	1	.	.	.	.	.	1	.	.	.	1	1	9
T_3	1	.	.	.	1	.	.	.	.	.	1	.	1	.	0	9
T_4	.	1	.	.	.	1	.	.	.	1	.	.	.	1	1	9
T_5	.	1	.	.	.	.	1	.	.	.	1	.	1	.	0	9
T_6	.	1	.	.	.	.	.	11	.	.	.	01	.	1	01	6,9

В общем случае количество ассерций, различающих n функциональных блоков, соединенных последовательно, равно n-1. Для решения задачи мониторинга программного кода в целях достижения максимальной глубины поиска дефектных модулей можно использовать таблицу активизации, по которой достаточно просто установить классы эквивалентных программных блоков, которым соответствуют одинаковые столбцы. В каждом классе, имеющем мощность, равную n блоков, необходимо задать n-1 ассерционный монитор для всех блоков, исключая последний из них. При этом таблица активизации становится неоднородной по размерности числа битов в каждой координате, которая определяется минимальным числом мониторов для распознавания программных блоков, соединенных последовательно. Кроме того, в таблице активизации появляется столбец ассерционных мониторов, задающий последовательность вершин графа для снятия состояния ассерций в процессе моделирования. В случае более чем 50% существенности всех ассерционных мониторов для диагностирования большинства блоков целесообразно сделать таблицу активизации однородной на полном полученном множестве ассерций. В данном случае полученная структура превращается в таблицу функций неисправностей (ТФН) [2], где точки в координатах обозначают нулевые векторы ассерционных состояний на одном тест-сегменте:

B_{ij}	B_1	B_2	B_3	B_4	B_5	B_6	B_7	B_8	B_9	B_{10}	B_{11}	B_{12}	B_{13}	B_{14}	V	A
T_1	101	.	101	.	.	.	.	.	001	.	.	.	001	.	000	3,6,9
T_2	001	.	.	001	.	.	.	.	.	001	.	.	.	001	001	3,6,9
T_3	001	.	.	.	001	.	.	.	.	.	001	.	001	.	000	3,6,9
T_4	.	001	.	.	.	001	.	.	.	001	.	.	.	001	001	3,6,9
T_5	.	001	.	.	.	.	001	.	.	.	001	.	001	.	000	3,6,9
T_6	.	101	.	.	.	.	.	010	.	.	.	001	.	001	001	3,6,9

Она предоставляет полную информацию о поведении программного продукта на тестовых сегментах, где реакции создаются с помощью минимального множества ассерционных мониторов, достаточного для распознавания любого программного модуля с функциональными нарушениями. Процесс-модель диагностирования сводится к одной итерации сравнения вектора экспериментальной проверки со столбцами матрицы активизации или ТФН:

$$L = L \underset{j=1}{\overset{n}{\vee}} B_j \leftarrow \sum_{i=\overline{1,k}}^{r=\overline{1,m}} (B_{ijr} \underset{i=1,k}{\overset{r=\overline{1,m}}{\oplus}} V_{ir}) = (0 \vee \min).$$

В данном примере использование приведенного выше критерия приводит к следующему результату: $L = B_{14} \leftarrow \forall i, r(B_{i,14,r} \oplus V_{i,r} = 0)$. Сигналы несовпадения, равные 1, отсутствуют при выполнении xor-операции между вектором V и столбцом 14 матрицы активизации. Таким образом, если задан тест уже имеется, то для повышения глубины диагностирования остается единственный путь, связанный с повышением количества ассерционных мониторов, которое увеличивает размерность матрицы активизации до таблицы функций неисправностей в соответствии с выражением: $Q = |T| \times |B| \times |A| = \text{card}T \times \text{card}B \times \text{card}A$.

Поэтому введение каждой новой ассерции должно иметь весомые аргументы, связанные с повышением глубины поиска дефектов, поскольку добавление каждой новой ассерции увеличивает размерность ТФН в два раза.

9. Модель эволюции киберпространства

Когда появляются новые свойства на структуре? Каково минимальное число компонентов для создания нового качества? Два – только на данном количестве компонентов можно создать новое свойство (property). Если нет второго элемента, его нужно придумать. В общем случае синтез модели (структуры) начинается с определения базовой функциональности, все остальные компоненты можно рассматривать как производные, присоединяемые посредством xor-операции

$$\{a\} \rightarrow \{a, b\} \rightarrow \{a, b, \oplus\} \rightarrow a \oplus b = c \rightarrow a \oplus b \oplus c = 0.$$

Чем может быть второй компонент по отношению к базе? Некоторым дополнением от противоположности до полного совпадения (рис. 11), которое дает возможность целенаправленно улучшить модель процесса или явления при наличии эталона, или при стремлении к нему (U – универсум примитивов):

$$b = \{\varnothing, a, U, a \cap b = a \wedge (a \neq b),$$
$$a \cap b = b \wedge (a \neq b), a \cap b \neq \varnothing \wedge (a \neq b \neq \varnothing),$$
$$\tilde{a},\ a \cap b = \varnothing\}.$$

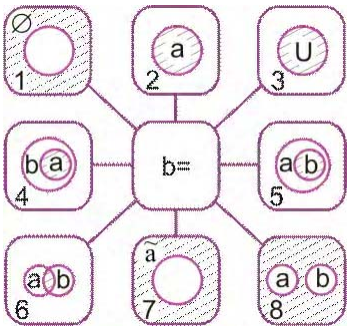


Рис. 11. Типы бинарного взаимодействия

Введение третьего и последующих компонентов должно быть аргументированным действием с позиции повышения адекватности модели или появления у нее новых свойств. Прокомментировать сказанное можно на примере графа технической диагностики $F \oplus T \oplus L = 0$. Базовый элемент – функциональность, производные от него – тест и список дефектов, каждый из которых может быть получен при условии, что

априори известно дополнение – дефекты или тест $T = F \oplus L,\ L = F \oplus T$. Триада минимальное число взаимодействующих объектов, образующее полезное новое свойство в форме транзитивного замыкания. Что есть базовый элемент в триаде, определяется субъективно и это не суть. Основа есть транзитивное замыкание любого числа компонентов посредством xor-операции. По существу графа технической диагностики – функциональность представлена таблицей истинности, тест есть сокращенная таблица истинности. Следовательно, xor-операция дает возможность формально отличить друг от друга два рассматриваемых компонента, в результате чего получится функциональное покрытие, выраженное в структурной оценке полноты теста относительно пространства состояний, заданного таблицей истинности. При задании дефектов и функциональности, взаимодействующих на основе xor-операции, получается тест, проверяющий все дефекты. Если к триаде компонентов добавить четвертую, в виде реального устройства, подлежащего тестированию, граф отношений $T \oplus F \oplus L \oplus U = 0$ становится более адекватным по сравнению с исходной моделью технической диагностики, поскольку возникает новое свойство, связанное с возможностью проведения диагностических экспериментов. При этом компонент – реальное изделие также имеет xor-взаимодействие со всеми вершинами графа. Взаимодействие оценивается структурным критерием качества бинарного xor-отношения.

Подводя некоторый итог сказанному выше, следует заметить, что на примере технической диагностики достаточно легко можно построить граф xor-отношений, включающий все известные технологии (модели и методы). Каждая новая вершина должна иметь ненулевую оценку взаимодействия относительно всех предыдущих $m \overset{n}{\underset{i=1}{\oplus}} A_i \neq 0$. В противном случае, она является повторением хотя бы одной из существующих и не может быть задекларирована как достижение или новшество. Таким образом, можно создать граф, содержащий тест, дефекты, функциональность, функциональное покрытие, реальное изделие, механизм ассерций $T \oplus L \oplus F \oplus C \oplus U \oplus A = 0$. Условием создания графа xor-взаимодействия компонентов является одинаковый формат данных. В противном случае необходимо создавать конверторы для переформатирования данных. Как следствие, из приведенных рассуждений можно создавать библиотеки типичных эталонов (прообразов) с производными от них в виде матриц отличий каждого образа от оригинала. В данном случае появляется возможность сокращения объема информации, подлежащей хранению, и быстрого воспроизведения полной модели любого их библиотечных образов. Сокращение возможно, если предположить, что критерии отличия образов от прообраза имеют преимущественно близкие к нулю оценки, для которых существуют эффективные методы сжатия информации. Для последовательностей двоичных строк в целях сжатия информации можно вычислять xor-отношение (расстояние) между соседними строками, что дает возможность получить матрицу расстояний, которая при последующей свертке имеет существенно большее число нулевых координат, что представляет интерес для средств кодирования информации в целях передачи данных, изображений и видеоинформации. Все сказанное относится только к моделям информационного взаимодействия процессов или явлений в киберпространстве. Что представляет собой производная на графах? Это – симметрическая разность между двумя графами (подграфами), представленными любой из известных форм (аналитическая, табличная, графическая), которая определяется применением xor-операции к двум компонентам $G_i \oplus G_j = (G_i \cap \overline{G}_j) \cup (\overline{G}_i \cap G_j)$.

Рассматривая в динамике развитие киберпространства (Второго мира), можно сделать экстраполирующий вывод, что оно стремится по меньшей мере повторить Первый (социум) во всех его отношениях, а затем и превзойти его по многим свойствам. Поэтому, чтобы ответить на вопрос, каким будет киберпространство как образ по взаимодействующим процессам или явлениям, необходимо посмотреть на прообраз, представленный системой взаимных отношений, существующих в Первом мире. Опять же понять, чего еще нет сейчас, но должно в качестве новых свойств (N) появиться во Втором мире, необходимо определить xor-отношение или сравнить два существующих мира $W_1 \oplus W_2 = N$.

10. Инфраструктура тестирования вредоносных компонентов и спама

На основе мультиматричного (-регистрового) процессора создана инфраструктура (рис. 12) тестирования электронной корреспонденции. Здесь фигурируют четыре процесс-модели: тестирование на стадии моделирования, диагностирование функциональных нарушений, оптимизация диагноза, восстановление работоспособности.

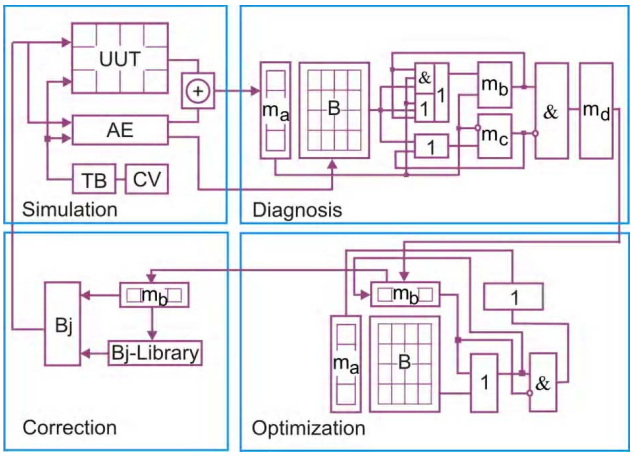


Рис. 12. Инфраструктура диагностирования электронной корреспонденции

1. Процесс-модель тестирования включает модель кода, механизм ассерций, testbench и coverage (по-

крытие). Последнее оценивает качество теста проверки всех состояний проекта. В результате моделирования синтезируется матрица активизации программных блоков B и матрица ассерционных реакций A на тестовые сегменты, которая может быть трансформирована к вектору состояния ассерций m путем применения функции or к вектор-столбцам A-матрицы:

$$\begin{cases} B = (T \oplus F); \\ m = \bigvee\limits_{j=1}^{m} A_j \leftarrow A = (T \oplus A^c). \end{cases}$$

1. Два последних компонента используются во второй процесс-модели для диагностирования блоков писем. Результатом диагностирования является вектор дефектов, формирующий подмножество блоков m_d с функциональными нарушениями. При этом не исключены ошибки как в testbench, так и в ассерционных операторах, отвечающих за тестирование и мониторинг программных блоков. Тройственная неопределенность диагноза $D = \{B_j, T_i, A_{ij}\}$ характерна при отсутствии точной идентификации блока в процедуре сравнения столбцов матрицы активизации с вектором ассерционных реакций.

2. Третий блок решает задачу минимизации числа блоков, подозреваемых в наличии функциональных нарушений, до одного из них. При этом используется матрица активизации блоков и диагноз m_d, полученный в предыдущей процесс-модели.

3. Модуль исправления функциональных нарушений ориентирован на ручной поиск ошибок в одном программном блоке, представленном вектором m_b. Возможен также автоматический режим исправления ошибок в блоках, если в инфраструктуре верификации предусмотрена библиотека диверсных программных модулей, имеющих аналогичные функциональности.

Предложенная инфраструктура является одним из шагов на пути создания автомата тестирования программных блоков. Далее представлен пример диагностирования функционального нарушения на основе использования матрицы активизации. Вектор ассерционных реакций получен из соответствующей матрицы $A_{ij} = \{1 \rightarrow \text{failed}, 0 \rightarrow \text{passed}\}$ путем дизъюнктивного объединения содержимого каждой строки:

$$m_i = \bigvee_{j=1}^{m} A_{ij} =$$

T	m
T_1	.
T_2	1
T_3	.
T_4	1
T_5	.
T_6	.
T_7	.
T_8	1

=

A_{ij}	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8
T_1	.	.	.	.	.	.	.	.
T_2	.	1	1	.	1	.	.	.
T_3	.	.	.	.	.	.	.	.
T_4	1	.	1	.	.	1	.	.
T_5	.	.	.	.	.	.	.	.
T_6	.	.	.	.	.	.	.	.
T_7	.	.	.	.	.	.	.	.
T_8	.	.	1	.	.	.	.	.

Последующее выполнение xor-операции между вектором ассерций и столбцами матрицы активизации блоков позволяет найти лучшее решение, которое определяется минимальным кодовым расстоянием

$$L = L \vee B_j \leftarrow \sum_{i=1}^{n} (B_{ij} \overset{n}{\underset{i=1}{\oplus}} m_i) = (0 \vee \min).$$

B_{ij}	B_1	B_2	B_3	B_4	B_5	B_6	B_7	B_8
T_1	1	.	.	1	.	.	1	.
T_2	.	1	1	.	1	.	.	.
T_3	.	.	.	.	.	1	1	1
T_4	1	.	1	.	.	1	.	.
T_5	.	1	.	1	.	.	.	1
T_6	1	.	.	.	.	.	1	1
T_7	.	1	.	.	1	1	.	.
T_8	.	.	1	1	1	.	.	.

$\oplus$

T	m
T_1	.
T_2	1
T_3	.
T_4	1
T_5	.
T_6	.
T_7	.
T_8	1

=

L_{ij}	B_1	B_2	B_3	B_4	B_5	B_6	B_7	B_8
T_1	1	.	.	1	.	.	1	.
T_2	1	.	.	1	.	1	1	1
T_3	.	.	.	.	.	1	1	1
T_4	.	1	.	1	1	.	1	1
T_5	.	1	.	1	.	.	.	1
T_6	1	.	.	.	.	.	1	1
T_7	.	1	.	.	1	1	.	.
T_8	1	1	.	.	.	1	1	1
$d(A, B_j)$	4	4	0	4	2	4	6	6

Результат диагностирования – блок 3 имеет функциональные нарушения, поскольку три ассерции «упали» на тестовых сегментах 2,4 и 8, которые в таком сочетании активизируют только блок с номером 3. Если выполнять процесс диагностирования, используя не вектор, а матрицу ассерций, то процесс поиска дефектных блоков будет иметь следующий вид:

B_{ij}	B_1	B_2	B_3	B_4	B_5	B_6	B_7	B_8
T_1	1	.	.	1	.	.	1	.
T_2	.	1	1	.	1	.	.	.
T_3	.	.	.	.	.	1	1	1
T_4	1	.	1	.	.	1	.	.
T_5	.	1	.	1	.	.	.	1
T_6	1	.	.	.	.	.	1	1
T_7	.	1	.	.	1	1	.	.
T_8	.	.	1	1	1	.	.	.

$\oplus$

A_{ij}	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8
T_1	.	.	.	.	.	.	.	.
T_2	.	1	1	.	1	.	.	.
T_3	.	.	.	.	.	.	.	.
T_4	1	.	1	.	.	1	.	.
T_5	.	.	.	.	.	.	.	.
T_6	.	.	.	.	.	.	.	.
T_7	.	.	.	.	.	.	.	.
T_8	.	.	1	.	.	.	.	.

=

L_{ij}	B_1	B_2	B_3	B_4	B_5	B_6	B_7	B_8
T_1	1	.	.	1	.	.	1	.
T_2	.	.	.	.	.	.	.	.
T_3	.	.	.	.	.	1	1	1
T_4	.	.	.	.	.	.	.	.
T_5	.	1	.	1	.	.	.	1
T_6	1	.	.	.	.	.	1	1
T_7	.	1	.	.	1	1	.	.
T_8	.	.	.	1	1	.	.	.
$d(A, B_j)$	2	2	0	3	2	2	3	3

Результат диагностирования аналогичен предыдущему – блок 3 имеет функциональные нарушения, поскольку кодовое расстояние равно нулю только для столбца с номером 3.

Практическая реализация моделей и методов тестирования была интегрирована в среду анализа Antispam компании Design and Test (рис. 13). Введенные в систему модули ассерций и библиотека писем (позитив, негатив, серая зона) существенно (на 10%) уменьшили серую зону, что дало возможность на 15% сократить общее время обработки корреспонденции.

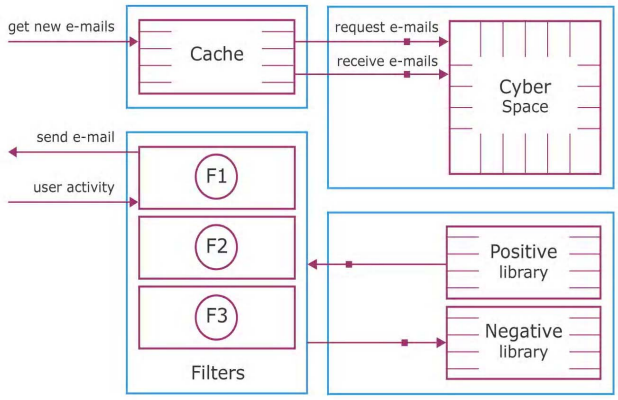
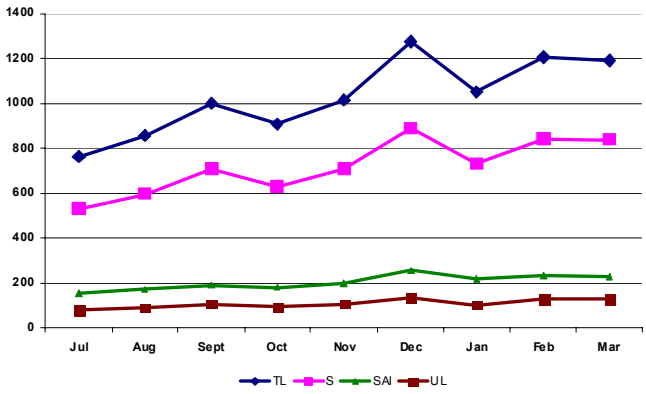


Рис. 13. Инфраструктура диагностирования спама

Фактически применение ассерций и библиотеки корреспонденции дает возможность каждому пользователю существенно (от 5 до 1 минуты) сократить время ручного анализа ежедневной корреспонденции (рис. 14). Механизм ассерций позволяет повысить качество диагностирования корреспонденции до уровня 97%. На рис. 14 приведены графики повышения качества анализа информационных сообщений по трем сегментам: позитив, негатив, серая зона, до и после внедрения анализатора Antispam на выборке из 100 пользователей за 9 месяцев.



	Jul	Aug	Sept	Oct	Nov	Dec	Jan	Feb	Mar
TL	764	854	1002	905	1015	1276	1052	1206	1193
S	532	595	706	630	707	889	731	844	836
SAI	152	170	190	180	202	254	219	233	226
UL	80	89	106	95	106	133	102	129	131

Рис. 14. Сравнительный анализ внедрения системы Antispam

Динамика занесения позитива в негативную библиотеку, равно как и наоборот, указывает на уменьшение процента неправильного распознавания по мере эксплуатации и обучения системы. Внедрение разработанного приложения в Украине, имеющей 12 млн пользователей почтовых сервисов, дает возможность получить следующий годовой эффект сэкономленного по стране времени в долларовом эквиваленте:

$$Q = N \times D \times T \times M =$$

$$= 12\ 000\ 000 \text{users} \times 250 \text{days} \times \frac{4}{60} \text{hours} \times \$5 =$$

$$= \$1\ 000\ 000\ 000$$

На рис. 15 представлен интерфейс (GUI), который даёт возможность пользователю работать со своей электронной почтой, которая была диагностирована с помощью системы Antispam. Нежелательные, неполезные письма были собраны в папке «Нежелательная почта», в то время как полезная почта сразу попадала в “INBOX” (папка с входящими сообщениями).

Рис. 15. Интерфейс системы Antispam

11. Интеллектуальный спам-фильтр. Описание системы Antispam

Каждое сообщение должно быть отнесено к одному из трех пулов (матриц): $L = \{L^0, L^1, L^x\}$ – позитивные, негативные, неопределенные (unknown state). Соответствующие им библиотеки $B = \{B^0, B^1, B^x\}$ имеют сигнатуры в виде двоичного вектора, который задает отношение принадлежности каждого сообщения к тестовым параметрам распознавания спама. В процессе обучения Antispam-системы формируется таблица функций спама (ТФС), открытая к пополнению:

$$B = \left|B_{ij}\right| = B \times T,$$

$$B = (B_1, B_2, ..., B_j, ..., B_n),$$

$$T = (T_1, T_2, ..., T_i, ..., T_m).$$

Для определения принадлежности нового сообщения m к одному из библиотечных типов сообщений L используется следующий критерий:

$$L = L \vee B_j \leftarrow \sum_{i=1}^{n}(B_{ij} \overset{n}{\underset{i=1}{\oplus}} m_i) = (0 \vee \min).$$

Он определяет кодовое расстояние путем последовательного сравнения между новым сообщением и каждым компонентом трех библиотек. Наличие простейшего мультиматричного процессора позволяет выполнять данные логические операции параллельно.

Стратегия (алгоритм) функционирования системы Antispam:

$$m = f(T) \rightarrow \begin{cases} L^0 = L^0 \vee B_i^0 \leftarrow B_i^0 \oplus m = 0; \\ L^1 = L^1 \vee B_i^1 \leftarrow B_i^1 \oplus m = 0; \\ L^x = L^x \vee B_i^x \leftarrow B_i^x \oplus m = (0 \vee \min). \end{cases}$$

1. Тестирование нового сообщения относительно существующих параметров спама. В результате каждое сообщение формализуется в вектор (сигнатуру) на множестве тестов:

$$m = f(T) = (m_1, m_2, ..., m_i, ..., m_m).$$

2. Определение принадлежности сообщения m пулу спама:

$$L^0 = L^0 \vee B_j \leftarrow \sum_{i=1}^{n}(B_{ij} \overset{n}{\underset{i=1}{\oplus}} m_i) = 0.$$

Если принадлежность сообщения к спаму не выявлена, переход к пункту 3, иначе – пополнение библиотеки спама, удаление его в корзину и переход к пункту 5.

3. Определение принадлежности сообщения m пулу позитива:

$$L^1 = L^1 \vee B_j \leftarrow \sum_{i=1}^{n}(B_{ij} \overset{n}{\underset{i=1}{\oplus}} m_i) = 0.$$

Если принадлежность сообщения к позитиву не выявлена, переход к пункту 4, иначе пополнение библиотеки позитива и переход к пункту 5.

4. Определение принадлежности сообщения m серому пулу:

$$L^x = L^x \vee B_j \leftarrow \sum_{i=1}^{n}(B_{ij} \overset{n}{\underset{i=1}{\oplus}} m_i) = (0 \vee \min).$$

5. Конец процедуры. Переход к пункту 1 для анализа нового сообщения.

Серый пул также имеет сигнатуры сообщений, но принадлежность к пулу определяется не только точным совпадением всех разрядов, но и минимальным кодовым расстоянием между сообщением и столбцом ТФС.

В процессе функционирования системы Antispam каждое серое сообщение проверяется на принадлежность к двум детерминированным пулам, которые постоянно совершенствуются путем модификации и пополнения критериев за счет учета опыта пользователя.

Тест-параметры представлены в таблице.

Пример диагностирования на основе использования таблицы функций спама представлен ниже, где на тест-параметрах имеется 6 переменных и 14 видов спама. Здесь вектор m есть новое сообщение, подлежащее идентификации.

B_{ij}	B_1	B_2	B_3	B_4	B_5	B_6	B_7	B_8	B_9	B_{10}	B_{11}	B_{12}	B_{13}	B_{14}	m
T_1	1	.	1	.	.	.	.	.	1	.	.	.	1	.	0
T_2	1	.	.	1	.	.	.	.	.	1	.	.	.	1	1
T_3	1	.	.	.	1	.	.	1	.	.	1	.	1	.	0
T_4	.	1	.	.	.	1	.	.	.	1	.	.	.	1	1
T_5	.	1	.	.	.	.	1	.	.	.	1	.	1	.	0
T_6	.	1	.	.	.	.	.	1	.	.	.	1	.	1	1

В соответствии с пунктом 2 алгоритма результат диагностирования – компонент 14:

$$L^0 = L^0 \vee B_{14} \leftarrow B_{14} \oplus m = 0,$$

дающий минимальное значение кодового расстояния, равное нулю.

Дополнительным пунктом повышения качества системы следует считать определение принадлежности автора спама к ближайшим «соседям» пользователя на основе анализа информации, полученной из Internet по запросу существования отношения «пользователь – источник». Это работает, если источник с пользователем имеет общие точки соприкосновения: публикации, конференции, университеты, школы, бизнес, отдых, интересы. В дальнейшем двоичные диаграммы решений как последовательная стратегия распознавания может быть использована для минимизации информационных массивов за счет усложнения структур данных.

Оценка качества модели диагностирования спама показывает эффективность применения пары (тест, ассерция – как реакция входного сообщения на тест-вопрос) для заданной глубины диагностирования. Оценка функционально зависит от длины теста $|T|$, числа ассерций наблюдения $|A|$, количества распознаваемых сообще-

ний N_d в матрице на общем числе примитивов, занесенных в библиотеку N:

$$Q = E \times D = \frac{]\log_2 N[}{|T| \times |A|} \times \frac{N_d}{N}.$$

Эффективность диагностирования есть отношение минимального числа двоичных разрядов, необходимых для идентификации (распознавания) всех примитивов к реальному количеству разрядов кода, представленному произведением длины теста на число ассерций в каждом из них. Если первая дробь оценки равна 1 и все примитивы матрицы распознаются ($N_d = N$), то тест и ассерции оптимальны, что доставляет критерию качества модели диагностирования значения, равного 1. Для матрицы B_{ij}, приведенной выше, оценка качества модели имеет вид:

$$Q = \frac{4}{6 \times 1} \times \frac{12}{14} = \frac{4}{7} = 0{,}57.$$

Оценка могла быть выше, если бы на тесте из шести вопросов распознавались сообщения B_3, B_9, которые являются эквивалентными.

12. Заключение

1. Предложена модель эволюционирующего кибернетического пространства, где субъектами выступают взаимодействующие процессы или явления с физическим носителем в виде компьютерных систем и сетей. Стандартизация пространства и всех взаимодействующих субъектов, включая негативные, возможна на основе предложенной бета метрики, которая структурированно и адекватно оценивает меру взаимодействия отношений в киберпространстве.

2. Дана универсальная неарифметическая модель структурной оценки отношения двух объектов в киберпространстве, которая способна распознавать любые типы теоретико-множественного или векторного взаимодействия объектов при решении практических задач распознавания образов и диагностирования технических изделий.

3. Разработаны инфраструктуры сервисного обслуживания программного продукта для защиты от вредоносных программ и верификации проектов цифровых систем, которая включает шесть компонентов, создающих условия для защиты и надежного функционирования программных и аппаратных компонентов.

4. Предложена новая модель программного продукта в форме графа блочных транзакций, а также новый матричный метод диагностирования функциональных нарушений, которые характеризуются технологичностью подготовки данных в процессе поиска некорректных блоков, что дает возможность существенно уменьшить время проектирования цифровых систем на кристаллах.

5. Усовершенствованы методы поиска функциональных нарушений, которые отличаются параллелизмом выполнения векторных операций над строками таблицы ФН, что дает возможность существенно (x10) повысить быстродействие вычислительных процедур, связанных с диагностированием и восстановлением работоспособности программных и аппаратных продуктов.

6. Разработана архитектура мультиматричного процессора, ориентированного на повышение быстродействия процедур встроенного диагностирования функциональных нарушений в программном или аппаратном изделии, которая отличается использованием параллельных логических векторных операций and, or, xor, slc, что дает возможность существенно (x10) повысить быстродействие диагностирования одиночных и/или кратных дефектов (функциональных нарушений).

Тест-параметры пулов

Позитивный пул	Негативный пул	Серый пул
1. Наличие в контактной книге владельца ящика 2. Известный RCPT 3. Авторизация отправителя 4. Наличие FQDN 5. Наличие отправителя 6. Содержание ASCII символов 7. Необходимое и допустимое количество команд в email 8. Наличие IP – адреса в белом локальном списке 9. Друзья ближайшего окружения по интересам 10. Вышестоящее начальство и подчиненные	1. Запрещённые символы в адресе 2. Если есть несовпадение значений команды HELO/EHLO 3. Отсутствие команды HELO 4. Отправители, представляющиеся как localhost.localdomain 5. Отправители, представляющиеся в HELO именем secondary mx или собственным IP 6. Отправители из dsl, dialup зоны в HELO 7. Отправители, представляющиеся именем локального компьютера 8. Пустой отправитель 9. Сброс по RCPT: неизвестный пользователь 10. Содержание не ASCII символов 11. Отправители, использующие IP адрес вместо домена 12. Отсутствие FQDN 13. Большое количество «сопроводительных» команд 14. Наличие словосочетаний из словаря 15. Наличие IP адреса в чёрных списках спам-листов (локальный, глобальный)	1. Возможность отписки 2. Неизвестный RCPT 3. Наличие большого количества получателей 4. Новый отправитель 5. Удаление предыдущих писем данного отправителя 6. Наличие словосочетаний из словаря 7. Наличие FQDN 8. Наличие отправителя 9. Содержание ASCII символов 10. Необходимое и допустимое количество команд в email 11. Наличие IP – адреса в белом локальном списке

7. Предложена инфраструктура диагностирования ВК программного кода цифровых систем, которая имеет четыре процесс-модели для тестирования, диагностирования, оптимизации и исправления ВК, замкнутые в цикл, что дает возможность уменьшить время сервисного обслуживания.

8. Разработана процесс-модель диагностирования спама в параллельном режиме на основе использования трех расширяемых матриц хранения и идентификации видов спама, что дает возможность существенно повысить качество диагностирования и уменьшить время установления диагноза входного сообщения.

8. Практическая реализация моделей и методов тестирования была интегрирована в среду анализа Antispam компании Design and Test. Введенные в систему модули ассерций и библиотека писем (позитив, негатив, серая зона) существенно (на 10%) уменьшили серую зону, что дало возможность каждому пользователю на 15% сократить общее время обработки корреспонденции.

Литература: 1. *Babulak E.* Future Global Office // 12th International Conference "Computer Modelling and Simulation". 2010. P. 352–356. **2.** *Caplan K., Sanders J.L.* Building an international security standard // IT Professional.– Vol. 1, Issue 2. 1999. P. 29-34. **3.** *Qishi Wu, Ferebee D., Yunyue Lin, Dasgupta D.* Visualization of security events using an efficient correlation technique // Computational Intelligence in Cyber Security, CICS '09. 2009. P. 61-68. **4.** *Инфраструктура* мозгоподобных вычислительных процессов / М.Ф. Бондаренко, О.А. Гузь, В.И. Хаханов, Ю.П. Шабанов-Кушнаренко. Харьков: Новое Слово. 2010. 160 с. **5.** *Хаханов В.И., Чумаченко С.В.* Модели пространств в научных исследованиях // Радиоэлектроника и информатика. 2002. №1. С. 124-132. **6.** IEEE Standard for Reduced-Pin and Enhanced-Functionality Test Access Port and Boundary-Scan Architecture IEEE Std 1149.7-2009. 985 p. **7.** *Da Silva F., McLaurin T., Waayers T.* The Core Test Wrapper Handbook. Rationale and Application of IEEE Std. 1500™. Springer. 2006. XXIX. 276 p. **8.** *Marinissen E.J., Yervant Zorian.* Guest Editors' Introduction: The Status of IEEE Std 1500.IEEE Design & Test of Computers. 2009. No26(1). P.6-7. **9.** IEEE Std 1800-2009 IEEE Standard for System Verilog-Unified Hardware Design, Specification, and Verification Language. http://ieeexplore.ieee.org/servlet/opac?punumber= 5354133. **10.** *Marinissen E.J.* Testing TSV-based three-dimensional stacked ICs // DATE 2010. 2010. P.1689-1694. **11.** *Benso A., Di Carlo S., Prinetto P., Zorian Y.* IEEE Standard 1500 Compliance Verification for Embedded Cores // IEEE Trans. VLSI Syst. 2008. No 16(4). P. 397-407. **12.** *Ubar R., Kostin S., Raik J.* Embedded diagnosis in digital systems // 26th International Conference "Microelectronics", MIEL 2008. 2008. P. 421-424. **13.** *Elm M., Wunderlich H.-J.* Scan Chain Organization for Embedded Diagnosis // Design, Automation and Test in Europe, DATE '08. 2008. P. 468–473. **14.** *Проектирование* и тестирование цифровых систем на кристаллах. Verilog & System Verilog / В.И. Хаханов, Е.И. Литвинова, О.А. Гузь.–Харьков: ХНУРЭ. 2009. 484с. **15.** *Проектирование* и верификация цифровых систем на кристаллах / В.И. Хаханов, И.В. Хаханова, Е.И. Литвинова, О.А. Гузь. Харьков: Новое слово. 2010. 528с. **16.** *Семенец В.В., Хаханова И.В., Хаханов В.И.* Проектирование цифровых систем с использованием языка VHDL. Харьков: ХНУРЭ. 2003. 492 с. **17.** *Хаханов В.И., Хаханова И.В.* VHDL+Verilog = синтез за минуты. Харьков: ХНУРЭ.2006. 264с. **18.** *Zorian Yervant.* Guest Editor's Introduction: Advances in Infrastructure IP // IEEE Design and Test of Computers. 2003. P.49-55. **19.** *Bulent I. Dervisoglu.* A Unified DFT Architecture for Use with IEEE 1149.1 and VSIA/IEEE P1500 Compliant Test Access Controllers. Proceedings of the Design Automation Conference. 2001. P. 53-58. **20.** *Bergeron J.* Writing Testbenches using SystemVerilog. Springer US.2006. 414 p. **21.** *Shibata T.* Implementing brain-like systems using nano functional devices // Ultimate Integration of Silicon, ULIS 2009. 2009. P. 131-134. **22.** *Soliman M.I., Al-Junaid A.F.* Codevelopment of Multi-level ISA and hardware for an efficient matrix processor // International Conference Computer Engineering & Systems.2009. P. 211-217. **23.** Senning C., Studer C., Luethi P., Fichtner W. Hardware-efficient steering matrix computation architecture for MIMO communication systems // IEEE International Symposium Circuits and Systems. 2008. P. 304-307. **24.** *Pedram Ardavan, Daneshtalab Masoud, Sedaghati-Mokhtari Nasser, Fakhraie Sied Mehdi.* A High-Performance Memory-Efficient ParallelHardware for Matrix Computation in Signal Processing Applications // International Symposium Communications and Information Technologies. ISCIT '06. 2006. P. 473-478. **25.** *Chenlong Hu, Ping Yang, Ying Xiao, Shaoxiong Zhou.* Hardware design and realization of matrix converter based on DSP & CPLD // 3rd International Conference Power Electronics Systems and Applications. 2009. P. 1-5. **26.** *Dave N., Fleming K., Myron King, Pellauer M., Vijayaraghavan M.* Hardware Acceleration of Matrix Multiplication on a Xilinx FPGA // 5th IEEE/ACM International Conference Formal Methods and Models for Codesign. 2007. P.97-100. **27.** *Loucks W.M., Snelgrove M., Zaky S.G.* A Vector Processor Based on One-BitMicroprocessors // IEEE Micro.1982. Vol. 2. Issue 1. P. 53-62. **28.** *Hilewitz Y., Lauradoux C., Lee R.B.* Bit matrix multiplication in commodity processors // International Conference Application-Specific Systems, Architectures and Processors. 2008. P. 7-12.

Поступила в редколлегию 06.04.2011

Рецензент: д-р техн. наук, проф. Филатов В.А.

Хаханов Владимир Иванович, декан факультета КИУ ХНУРЭ, д-р техн. наук, профессор кафедры АПВТ ХНУРЭ. Научные интересы: техническая диагностика цифровых систем, сетей и программных продуктов. Увлечения: баскетбол, футбол, горные лыжи. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 70-21-326. E-mail: hahanov@kture.kharkov.ua.

Чумаченко Светлана Викторовна, заместитель декана факультета КИУ ХНУРЭ, д-р техн. наук, профессор кафедры АПВТ ХНУРЭ. Научные интересы: математическое моделирование, методы дискретной оптимизации. Увлечения: фотография. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 70-21-326. E-mail: ri@kture.kharkov.ua.

Литвинова Евгения Ивановна, заместитель декана факультета КИУ ХНУРЭ, д-р техн. наук, профессор кафедры АПВТ ХНУРЭ. Научные интересы: техническая диагностика цифровых систем. Увлечения: плавание, горные лыжи. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 70-21-421. E-mail: kiu@kture.kharkov.ua.

Мищенко Александр Сергеевич, аспирант кафедры АПВТ ХНУРЭ. Научные интересы: технологии тестирования и WEB-приложения. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 70-21-326. E-mail: alex@simplesolutions.com.ua.

Адамов Александр Семенович, преподаватель ЦНСИМ ХНУРЭ, аспирант кафедры АПВТ ХНУРЭ. Научные интересы: антивирусные технологии,технологии тестирования и WEB-приложения. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 70-21-326. E-mail: alex@simplesolutions.com.ua.