Перечислим основные этапы обработки. При этом предположим, что СВ состоит из ПЭВМ и спецпроцессора, который, как минимум, включает в себя подсистему геометрической обработки (ПГО) в соответствии с [3] и рассматриваемую в данной статье подсистему обработки метеоусловий (ПОМ).

1. ПЭВМ для каждого кадра изображения вычисляет пару уменьшаемых в соотношениях (13) — (15) в зависимости от случаев 1, 2, 3, которые заносятся в ПОМ, где хранятся в течение обработки кадра изображения. В ПОМ заносится также параметр $I\infty$.

2. На каждом такте работы конвейера в ПОМ из ПГО поступает $\log_2 V_y$.

3. Вычисляется $\log_2\rho$ в соответствии с (13)-(15).

4. Табличным методом вычисляется коэффициент пропускания τ (7).

5. Одновременно вычисляются составляющие цвета (9). Умножения в (9) выполняются табличным методом.

В заключение отметим, что приведенная математическая модель и алгоритм обработки метеоусловий использованы при построении СВ, образцы которой установлены на комплексный тренажер в КТС ТУ-154М в учебно-сертификационном центре гражданской авиации в г. Киеве, а также в Харьковском институте летчиков на летный тренажер ТЛ-39.

Список литературы: 1. *Электросветосигнальное* оборудование аэродромов / Фрид Ю. В., Величко Ю. К., Козлов В. Д. и др. М.: Транспорт, 1988. 318 с. 2. *Мартинес Ф.* Синтез изображений: Пер. с франц. М.: Радио и связь, 1990. 192 с. 3. *Гусятин В.М.* Алгоритм геометрических преобразований изображения в растровых видеосистемах реального масштаба времени // Авиационно-космическая техника и технология. Харьков, 1998.

Поступила в редколлегию 12.11.99

Гусятин Владимир Михайлович, канд. техн. наук, доцент кафедры электронных вычислительных машин ХТУРЭ. Научные интересы: теория и практика построения спецпроцессоров растровых графических систем реального времени. Адрес: Украина, 61726, Харьков, пр. Ленина, 14, тел. 40-93-54, 66-61-22.

Остроушко Андрей Павлович, аспирант кафедры ЭВМ ХТУРЭ. Научные интересы: растровые графические системы реального времени. Адрес: Украина, 61168, Харьков, пр. Ленина, 14, тел. 40-93-54.

УДК 681.325:519.713

В.И. ХАХАНОВ, Е.В. КОВАЛЕВ, В.В. ХАНЬКО, МАСУД МД. МЕХЕДИ

СИСТЕМА ГЕНЕРАЦИИ ТЕСТОВ ДЛЯ ПРОЕКТИРОВАНИЯ ЦИФРОВЫХ АВТОМАТОВ В СРЕДЕ VHDL-ACTIVE

Предлагается система генерации тестов для конечных автоматов, описываемых в среде VHDL-Active [1]. Исходная информация об объекте диагностирования задается в виде графа переходов или в формате конструкций языков описания аппаратуры VHDL, Verilog. Приводится описание системы генерации тестов ASFTEST с позиции пользователя. Система может быть полезна для проектировщиков цифровых управляющих автоматов, в качестве средства верификации созданного устройства.

1. Постановка задачи

Современные системы проектирования цифровых устройств, как правило, ориентируются на быстродействующие программируемые интегральные схемы (ПЛИС) – Field Programable Gate Array (FPGA), Complex Programable Logic Device (CPLD). К их достоинствам относятся сравнительно невысокая стоимость, минимальное время проектирования устройства, высокое быстродействие выполнения микроопераций.

Процессу проектирования и синтеза цифрового автомата предшествует достаточно трудоемкая процедура построения или описания модели на языках VHDL, Verilog. В качестве альтернативного способа описания цифрового автомата предлагается визуальный интерфейс создания рисунков графа переходов или структуры примитивных элементов. Далее выполняется компиляция внутренних структур данных в целях моделирования и верификации цифрового проекта. Однако для упомянутой процедуры необходимы тестовые последовательности, генерация которых представляет достаточно сложную задачу. В зависимости от сложности и функций проекта применяются несколько основных методов генерации тестов: исчерпывающий, псевдослучайный, алгоритмический.

Первый ориентирован на полную проверку цифрового устройства на всех 2^{n+m} входных наборах (n – число входных переменных, m – количество внутренних состояний). Метод прост в реализации, имеет 100%-ное качество покрытия неисправностей, но требует значительных временных затрат в процессе тестирования устройства.

Второй метод использует псевдослучайные генераторы (сигнатурные анализаторы) для получения тестовых последовательностей. Он имеет максимальное быстродействие, но качество теста может быть неприемлемо низким для последовательностных схем.

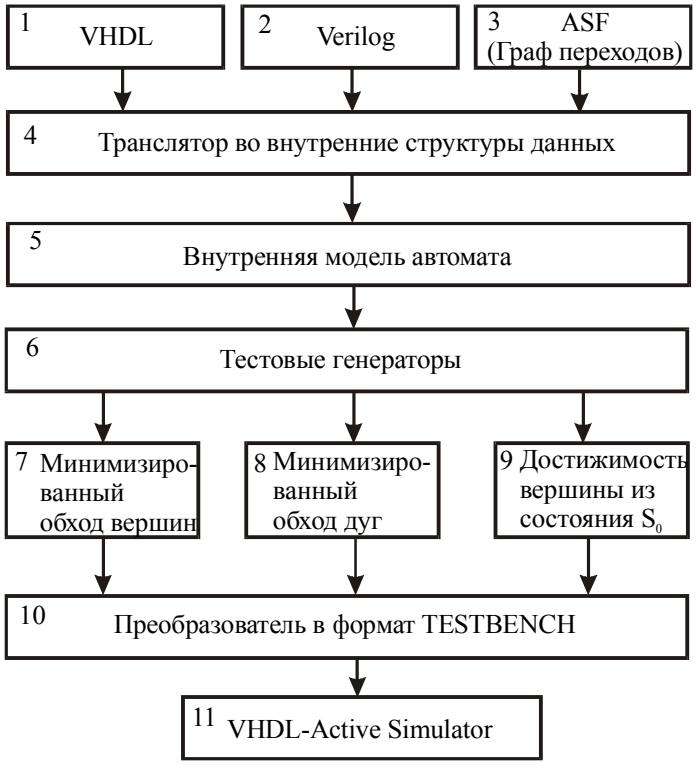

Рис. 1

Третий ориентирован на проверку конкретных типов неисправностей, как правило одиночных константных, построением логических путей активизации функций или переменных. Получаемый тест компактен, но для его проектирования требуется много времени. Решение данной проблемы в рамках создания тестового генератора для конечных автоматов предлагается ниже.

Цель создания программы ASFTEST – автоматизация проектирования теста проверки исправного поведения и неисправностей конечного автомата, заданного в форме графа переходов. При этом считается, что построенный тест проверяет практически все одиночные константные неисправности [2, 3] реализуемого на основе ПЛИС устройства. Структура программного обеспечения генератора тестов представлена на рис. 1.

2. Задачи, решаемые программой ASFTEST

1. Синтаксический и семантический контроль (блок 4, рис.1) исходного описания конечного автомата, заданного в форме графа переходов (описания на языке VHDL или Verilog) (блоки 1-3) с последующей трансляцией языковых конструкций во внутренние структуры данных (блок 5) программы ASFTEST.

2. Генерация трех типов тестов (блок 6) на основе использования метода ветвей и границ для обхода всех вершин (дуг) графа переходов автомата:

– тест минимального обхода всех вершин графа переходов конечного автомата (блок 7);

– тест минимального обхода всех дуг графа переходов конечного автомата (блок 8);

– тест достижимости каждой вершины графа из начального состояния с последующим возвратом в него (блок 9).

3. Форматирование теста по стандарту файла описания тестов (Testbench) системы проектирования VHDL-Active (блок 10).

Форматированный тест далее поступает на систему моделирования пакета VHDL-Active, который предназначен для анализа исправного поведения проектируемого цифрового автомата. Упомянутая система имеет визуальный интерфейс просмотра временных или цифровых диаграмм каждой линии цифрового устройства. Решение о правильности его работы в конечном счете принимает разработчик.

3. Теоретическое обоснование программы ASFTEST

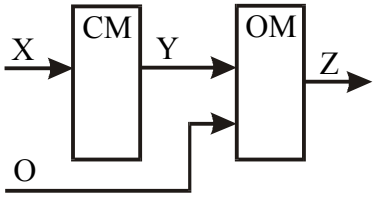

Рис. 2

Любой цифровой проект является конечным автоматом. Теоретически в процессе проектирования необходимо разделять автоматы на управляющую и операционную части: SM={CM, OM}. При этом должно выполняться условие взаимодействия автоматов в соответствии с рис.2. В этом случае управляющий автомат представлен множеством $X = \{X_1,...,X_i,...,X_k\}$, внутренних $S = \{S_1,...,S_j,...,S_m\}$ и выходных $Y = \{Y_1,...,Y_r,...,Y_n\}$ состояний. Каноническое задание управляющего автомата определяется функциями переходов и выходов автомата первого рода: $S_t = f(X_t, S_{t-1})$; $Y_t = g(X_t, S_{t-1})$, где $t-1, t$ – фреймы автоматного времени.

В процессе проектирования конечного автомата разработчик не заботится о строгом разделении на управляющую и операционную части. Его цель – уменьшить время проектирования и аппаратурные затраты автомата в целом. Для этого используются дополнительные переменные памяти в виде оповестительных сигналов. В этом случае появляется структура отношений, показанная на рис.3. Здесь имеется зависимость управляющего автомата от состояний оповестительных сигналов $Z = (Z_1, ..., Z_s, ..., Z_p)$ операционного автомата. Это противоречит каноническому заданию управляющего автомата. Поэтому возникает необходимость в модификации модели конечного автомата, которая представлена на рис.4. Функции управляющего автомата определяются уравнениями

$S_t = f(X_t, S_{t-1}, Z_{t-1})$; $Y_t = g(X_t, S_{t-1}, Z_{t-1})$.

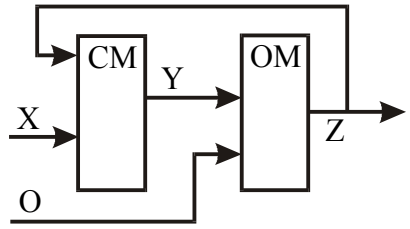

Рис. 3

В общем случае операционный автомат по способу описания не отличается от управляющего. Распределение ролей между двумя взаимосвязанными автоматами определяется наличием

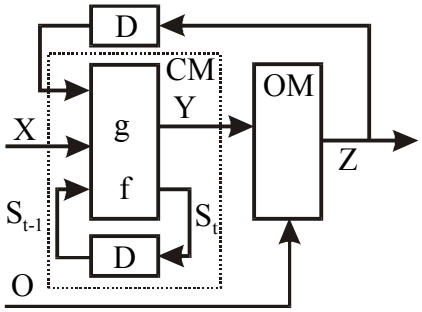

Рис. 4

входных сигналов. Если последние присутствуют в обеих структурах, установить роль ведущего и ведомого достаточно сложно, поскольку любой автомат может быть описан графом переходов автомата. Но классификация схемных стандартов операционного автомата по их функциям (триггеры, счетчики, регистры, память, коммутаторы, декодеры, АЛУ) позволяет оперировать ими как примитивами. В этом случае управляющий автомат рассматривается как способ описания взаимодействия примитивов операционного автомата во времени. Специалист в области цифрового проектирования должен уметь оперировать готовыми решениями на уровне алгоритмических примитивов. По-видимому, разделение конкретного проекта на управляющий и операционный автоматы субъективно и является следствием практического опыта и квалификации разработчика.

Следует подчеркнуть, что функциональных свойств классических автоматов Мура или Мили часто не хватает для оптимальной реализации управляющих воздействий – функций выходов автомата. Поэтому в системе VHDL-Active предусмотрен следующий формат описания таблицы переходов графа автомата:

$$<X,\ Y,\ Z^{Y},\ Z^{XY},\ Z^{XY+},\ Z^{+XY}>,$$

где $<Z^{Y},\ Z^{XY},\ Z^{XY+},\ Z^{+XY}>$ – четыре операции, которые могут выполняться для каждой пары <X, Y,>. Акция Z^{Y} соответствует состоянию автомата (вершине графа переходов); Z^{XY} – отождествляется со средней частью перехода; Z^{XY+}– с окончанием перехода; Z^{+XY} – с началом перехода.

3.1. Модель функции возбуждения. Существует проблема с получением теста обхода всех вершин и дуг графа переходов, связанная с решением задачи установки условий перехода: $(X_j, S_i, Y_j) \Rightarrow (S_i \to S_j)$. Формально упомянутая задача сводится к обратной импликации на функции возбуждения $F_{ij} = (X_j, S_i, Y_j)$ $(F_{ij} = \{0,1\})$ для рассматриваемого перехода $S_i \to S_j$. Если значение $F_{ij} = 1$, то переход выполняется, в противном случае ($F_{ij} = 0$) – нет. В общем случае входная переменная $U_r \in (X_j, S_i, Y_j)$ определяется типами: $U_r = \{B, I, V, L\}$, где B – булева переменная; I – целого типа в интервале $(0 - 2^{16})$; V – векторная переменная длиной (1 – 32); L – логическая многозначная переменная (для стандарта VHDL – 9 символов). Для небулевых переменных записывается предикат, который принимает истинное или ложное значение, что записывается в формате таблицы истинности. Таким образом, функция возбуждения

сводится к структуре типов функций, каждая из которых на выходе имеет двоичное значение. Это означает, что к такой схеме можно применить модифицированный П-алгоритм [2], который по единичному состоянию функции возбуждения может определить решения в виде входных сигналов, обеспечивающих требуемый переход $S_i \to S_j$.

Пример. Записать в виде функции возбуждения следующее условие перехода: если A больше 20 и C равно переднему фронту (E) или D=1, то выполняется переход $S_1 \to S_2$. Отсутствие переднего фронта обозначается символом F. Значения переменной целого типа обозначать минимальным количеством интервалов.

Таблица истинности такого предиката имеет следующий вид:

A_I	C_L	D_B	F
$21-2^{16}$	E	X	1
X	X	1	1
$0-20$	X	1	0
X	F	1	0
X	X	0	0

.

Схема, составленная из подобных примитивов, обрабатывается модифицированным П-алгоритмом, в результате чего находится решение, удовлетворяющее условию перехода. Кроме того, такие примитивы могут составлять многоярусную сколь угодно сложную схему, поскольку выходные значения элементов определены в двоичном алфавите. Очевидно, что в такой схемной структуре переменные типа $\{I, L\}$ могут быть только внешними входами, что упрощает обратную импликацию и делает ее практически на модели двоично-определенной функциональной структуры.

3.2. Алгоритмы тестирования автомата. Алгоритмический уровень описания цифрового проекта является предпочтительным для решения проблемы генерации тестов исправного поведения. Это определяется:

1) доступностью первоначальных знаний о поведении объекта;

2) приведением проблемы генерации тестов к минимальному обходу вершин (переходов) в графе;

3) меньшим количеством состояний абстрактного автомата. Аппаратурная реализация последнего не может быть выполнена оптимально, без избыточности;

4) более технологичными и простыми алгоритмами генерации тестов. Это связано с меньшими объемами информации описания объекта на алгоритмическом уровне по сравнению с вентильным или функциональным.

Однако и в случае алгоритмического описания конечного автомата существует проблема обеспечения всех условий выполнения переходов. Активизация перехода S_i-S_j определяется булевой функцией: F = f(X, Z). Наличие оповестительных сигналов Z в условии перехода предполагает не только знание конкретного Z_i, но и априорное получение

требуемого для перехода состояния Z_i. Для решения упомянутых задач следует:

1. Преобразовать неявное (аналитическое) описание примитивов ОМ к их явной форме задания (таблицы переходов, граф переходов, кубическое покрытие). Преимущества упомянутой формы заключаются в возможности выполнения прямой и обратной импликаций.

2. В пространстве и во времени выполнять обратную импликацию условий перехода Z_i. Это есть NP-сложная задача технической диагностики, которая в общем случае не имеет практически приемлемого решения. Одно из рациональных решений задачи установки состоит в псевдослучайной генерации установочной последовательности с проверкой на каждом входном наборе Z_i выполнения условий Z_i; использованием процедуры моделирования. Другое решение связано с <u>наличием</u> описания операционного автомата или его части в виде графа переходов. В этом случае несложно найти путь для достижения требуемого состояния Z_i методом обратного прослеживания переходов автомата.

3. Выполнять прямую импликацию входных условий для управляющего и операционного автомата в целях верификации сгенерированного теста обхода всех дуг (вершин) графа. Для этого необходимо иметь автономную систему исправного моделирования.

Рассмотрим основные подходы к тестированию конечных автоматов, заданных графами переходов.

Мощность множества всех состояний автомата определяется выражением $Q = \mathrm{card}(S) \times \mathrm{card}(Y) \times \mathrm{card}(X)$. Создание теста для конечного автомата заключается в генерировании входных векторов, число которых равно Q. Тест должен содержать для каждой достижимой пары непротиворечивых состояний $(S_j, Y_r) \in (S \times Y)$ наборы $\overset{k}{\underset{i=1}{\forall}}(X_i \in X)$. Учитывая, что не все триады $(X_i, S_j, Y_r) \in (X \times S \times Y)$ совместимы, фактическая мощность теста проверки исправного поведения для реальных автоматов может быть меньше оценки Q в сотни раз. В отличие от комбинационных автоматов, где тест проверки исправности есть таблица истинности, имеющая размерность $Q^k = 2^p$ (p – число булевых входных переменных), для последовательностных устройств генерация теста с верхней оценкой Q является рациональным решением. Проектирование теста проверки исправности основано на следующих стратегиях:

1. Построение теста минимального обхода всех вершин графа переходов конечного автомата $X = \min\limits_{X}[\overset{n}{\underset{j=0}{\forall}} X_j(S_j)]$. Проверяются одиночные неисправности переходов (ОНП): $S_i \Rightarrow S_j / S_i \Rightarrow S_r (j \neq r)$, когда вместо перехода $S_i \Rightarrow S_j$ выполняется $S_i \Rightarrow S_r$. Не проверяются одиночные константные неисправности (ОКН), связанные с булевыми переменными установки и синхронизации конечного автомата.

2. Построение теста достижимости каждой вершины графа из начального состояния с последующим возвратом автомата в S_0: $X = \overset{n}{\underset{j=1}{\forall}}[X_j(S_0 \Rightarrow S_j \Rightarrow S_0)]$. В дополнение к тесту, созданному по 1-стратегии, проверяются логические функции установки автомата из произвольного состояния.

3. Построение теста минимального обхода всех дуг графа переходов конечного автомата $X = \overset{i=\overline{0,n}}{\underset{j=\overline{1,n-1}}{\forall}}[\forall X_{ij}(S_i \Rightarrow S_j)]$. Допускается наличие в графе более чем одной дуги $S_i \Rightarrow S_j$. Проверяются ОНП, а также исправность функций автомата, обеспечивающих все переходы.

4. Построение теста обхода всех дуг графа конечного автомата $X = \overset{i=\overline{0,n}}{\underset{j=\overline{0,n}}{\forall}}\{\forall X_{ij}[\forall(S_i \Rightarrow S_j)]\}$ реализацией всех условий каждого перехода и сохранения состояния. Обеспечивается проверка всех существенных ОКН и ОНП, а также полная проверка исправности конечного автомата. Длина теста максимально приближена к оценке Q. Создание такого теста – идеальное решение проблемы не только тестирования реальных цифровых автоматов, но и его формальной верификации.

4. Заключение

В качестве областей применения системы ASFTEST можно рассматривать следующее:

1. Тестирование цифрового автомата в целях верификации его функций на стадии проектирования.

2. Использование теста цифрового автомата для проверки его исправности на стадии эксплуатации.

3. Использование теста цифрового автомата для построения алгоритмов поиска дефектов и их диагностирования на стадиях проектирования и эксплуатации.

Программа генерации тестов ориентирована на интеграцию в системы проектирования фирм: Aldec, Xilinx. Операционные системы, поддерживающие ASFTEST: Windows NT, Windows 98. Объем программы – 8 тыс. операторов языка C++. Время генерации тестов – от нескольких секунд до 1 часа. Количество обработанных примеров – 120 со средним число состояний 40. Среднее время построения тестов 15 с.

Список литературы: 1. *Active*-VHDL Series. Book #1 - #4. Reference Guide. ALDEC Inc. 1998. 206 p. 2. *Хаханов В.И.* Техническая диагностика элементов и узлов персональных компьютеров.К.: ІЗМН. 1997. 308 с. 3. *Abramovici M., Breuer M.A. and Friedman A.D.* Digital System Testing and Testable Design, Computer Science Press, 1998. 652 p.

Поступила в редколлегию 14.06.2000

Хаханов Владимир Иванович, д-р техн. наук, профессор кафедры автоматизации проектирования вычислительной техники ХТУРЭ. Научные интересы: техническая диагностика вычислительных устройств, систем, сетей. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 40-93-26. E-mail: Hahanov@kture.kharkov.ua

Ковалев Евгений Викторович, аспирант кафедры автоматизации проектирования вычислительной техники ХТУРЭ. Научные интересы: техническая диагностика компьютерных систем и сетей. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 40-93-26.

Ханько Вадим Викторович, аспирант кафедры автоматизации проектирования вычислительной техники ХТУРЭ. Научные интересы: техническая диагностика компьютерных систем и сетей. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 40-93-26.

Масуд МД. Мехеди, аспирант кафедры автоматизации проектирования вычислительной техники ХТУРЭ. Научные интересы: техническая диагностика вычислительных систем. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 40-93-26.

УДК 519.7

Н.А. ЯКИМОВА

АЛГЕБРАИЧЕСКИЙ СИНТЕЗ ПРОСТЫХ СЛОВОСОЧЕТАНИЙ В ПРЕДЛОЖЕНИЕ ЕСТЕСТВЕННОГО ЯЗЫКА

Для естественно-языковой интерпретации элементов векторного логического пространства разрабатывается методика алгебраического синтеза простых словосочетаний в повествовательное распространенное предложение естественного языка, которая не зависит от конкретного языка, так как основывается на семантически универсальных описаниях объектов окружающего мира.

Рассмотрим сложное словосочетание [1]. Например, "очень большой ярко-синий цветок". Обозначим его через S. Построим схему связей слов в этом словосочетании, указав стрелками направление связи от подчиненного слова к главному (рис. 1):

ЯРКО $\rightarrow$ СИНИЙ $\rightarrow$ ЦВЕТОК $\leftarrow$ БОЛЬШОЙ $\leftarrow$ ОЧЕНЬ
2 1 1 2

Рис. 1. Схема связей слов в сложном словосочетании S

На схеме цифрой "2" обозначен второй, а цифрой "1" – первый уровни связей. При записи выражения для математической реализации рассматриваемого словосочетания будем двигаться по схеме от подчиненного слова к главному, разбив для этого словосочетание на синтагмы таким образом, чтобы главное слово каждого последующего уровня было подчиненным словом предыдущего. Таким образом, получаем синтагмы "ярко→синий", "синий→цветок", "очень→большой", "большой→цветок".

Рассмотрим первый уровень связи. Как было показано в [2], выражение "большой синий цветок" формально можно записать как ab$l$. Пусть вектор l является некоторым вектором $Q_i^1(x, y)$, i=0, ..., 15 пространства $\boldsymbol{Q}_1$ двухместных предикатов над скалярным полем $\boldsymbol{P}$ одноместных предикатов. Тогда скалярам a и b соответствуют некоторые одноместные предикаты $P_j(x)$ и $P_t(x)$, j, t=0, ..., 3. Рассмотрим теперь синтагмы "ярко→синий" и "очень→большой". Если наречиям "очень" и "ярко" поставить в соответствие скаляры k_p и k_q из двухэлементного множества K={0, 1}, то предикаты $P_j(x)$ и $P_t(x)$ можно рассматривать как векторы логического пространства, заданного над скалярным полем K, т.е. эта синтагма формализуется как другая алгебра, но тоже логического типа. Этот вариант формализации представлен на рис. 2,а:

очень (k_p) → большой $(P_j(x))$ →цветок $(Q_i(x, y))$← синий $(P_t(x))$←ярко (k_q)

а

очень $(Q_r^2(x,y))$→большой $(P_j(x))$→цветок $(Q_i(x, y))$←синий $(P_t(x))$← ярко $(Q_f^2(x, y))$

б

очень $(P_r^2(x))$→большой $(Q_j^1(x,y))$→цветок $(P_i^1(x))$←синий $(Q_f^1(x,y))$←ярко $(P_t^2(x))$

в

очень $(Q_t(x, y))$→большой $(P_j(x))$→цветок (k_p)←синий $(P_i(x))$←ярко $(Q_r(x, y))$

г

Рис. 2. Варианты формализации сложного словосочетания S

Поскольку формализацию можно проводить в любом направлении [2], формализуя синтагмы "очень→большой" и "ярко→синий", предикаты $P_j(x)$ и $P_t(x)$ можно рассматривать как элементы скалярного поля, над которым задано также пространство $\boldsymbol{Q}_2$ двухместных предикатов, векторы $Q_r^2(x, y)$ и $Q_f^2(x, y)$, r, f=0, ..., 15 которого соответствуют наречиям "очень" и "ярко". Этот вариант формализации представлен на рис. 2,б. Однако существует и третий вариант формализации словосочетания S. Если формализовать 2-й уровень связи в направлении от подчиненного слова к главному, а 1-й – в обратном направлении, то объекту "цветок" можно поставить в соответствие одноместный предикат $P_i^1(x)$, являющийся элементом скалярного поля $\boldsymbol{P}_1$, над которым задано логическое пространство $\boldsymbol{Q}$ двухместных предикатов, векторы $Q_j(x, y)$ и $Q_f(x, y)$, j, f=0, ..., 15 которого отвечают прилагательным "большой" и "синий" соответственно. В то же время набор векторов $\boldsymbol{Q}$ можно рассматривать как логическое пространство над скалярным полем одноместных предикатов $\boldsymbol{P}_2$, элементы которого $P_r^2(x)$ и $P_t^2(x)$, r, t=0, ..., 3 соответствуют наречиям "очень" и "ярко". Этот вариант формализации представлен на рис. 2,в.

Последний вариант формализации словосочетания S предполагает формализацию всех уровней связи в направлении от главного слова к