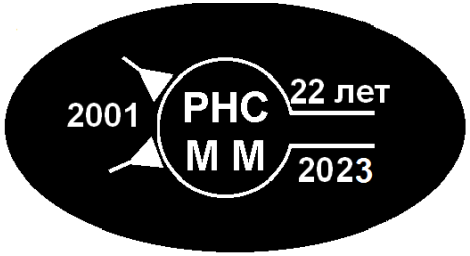
Республиканский научный семинар
«МЕТОДЫ МОДЕЛИРОВАНИЯ»



В.А. Райхлин (КНИТУ-КАИ)

ПРЕДСТАВЛЕНИЕ К ПУБЛИКАЦИИ МОНОГРАФИИ:

Райхлин В.А.,Морозов А.В.,Валиуллина Л.Р.,Фадеев К.А.
КОНСТРУКТИВНОЕ МОДЕЛИРОВАНИЕ ЦИФРОВЫХ АВТОМАТОВ */Под ред. В.А. Райхлина*

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Функционирование цифрового автомата обычно задается на практике рекурсивно (неформально), в виде множества правил. Задача распознавания представимости такого задания конечным автоматом и его синтеза алгоритмически неразрешима и требует привлечения элементов эвристики. Решением этой задачи ранее занимался ряд ученых. Однако развитые ими подходы нельзя отнести к числу детерминированных. Их практическое применение вызывало серьезные трудности.

В основе развитого нами подхода – методология конструктивного моделирования систем КМС. Процесс такого моделирования позволяет выявить некоторые свойства эффективных реализаций системы. В силу аксиомы знания модальной логики («что известно, то верно») они постулируются как закономерности. Использование в качестве средств доопределения задач выверенных постулатов как основ теории делает найденный конструктивный метод адекватным решаемой задаче на текущий период времени.

КМС занимается вопросами синтеза сложных систем в условиях неполноты информации. Недостаток информации восполняется открытиями по результатам эксперимента, которые не поддаются строгому доказательству. Надо иметь мужество декларировать эти открытия как конструктивные закономерности.

«Опыт есть единственный непреложный аргумент, и как бы ни было удивительно то, что он нам говорит, мы обязаны ему верить и строить новые теории применительно к тому, что мы видим, не смущаясь противоречиями со старым и привычным» **.**

Н.Н. Семенов, творец теории цепных реакций.

В монографии систематизированы результаты многолетних исследований авторского коллектива по различным вопросам конструктивного моделирования цифровых автоматов. Большинство из них было ранее опубликовано в различных изданиях. Но мы полагаем полезной предпринятую систематизацию для привлечения внимания «широкого круга» к перспективным, на наш взгляд, подходам.

Книга включает 6 разделов и 2 приложения. Раздел I – развернутый очерк методологии конструктивного моделирования систем. В разделе II на основе постулируемого объектного определения состояний автомата развивается эвристический подход к синтезу автоматов по неформальному заданию. Он достаточно детерминирован. Раздел III раскрывает трудности построения последовательностных схем по найденной таблице переходов. Раздел IV посвящен вопросам автоматизации процесса синтеза. Вопросы разработки соответствующей интерактивной системы рассматриваются в Разделе V. От трудностей абстрактного и структурного синтеза автоматов избавляет переход к программной модели, рассматриваемый в Разделе VI. С ним связывается перспектива применения автоматных подходов к защите беспилотных летательных аппаратов в экстремальных ситуациях.

Введение, разделы I–IV и оба приложения написаны В.А. Райхлиным. Раздел V – А.В. Морозовым. Раздел VI – Л.Р. Валиуллиной и К.А. Фадеевым.

I. ШТРИХИ МЕТОДОЛОГИИ КМС

Главное утверждение этой методологии:

Основой развития любого научно-технического направления являются выверенные идеи, которые постулируются как закономерности.

II. ОБЪЕКТНАЯ МОДЕЛЬ СИНТЕЗА АВТОМАТОВ

Теорема 2.1. Если автомат Мили синтезирован таким образом, что переход в любое его состояние s^k происходит только при одном значении выхода z^k, то каждое состояние автомата может быть специфицировано некоторой группой элементов события, отмеченного соответствующим значением выхода.

Ограничение 2.1. Переход в любое s^k происходит только при одном изменении входа.

Постулат 2.1. ***При выполнении условий теоремы 2.1 и ограничения 2.1, внутреннее состояние автомата – объект, специфицированный кортежем $<x^{k-1} – x^k, z^k, J^k>$. Здесь $(x^{k-1} – x^k)$ – изменение входа, при котором в такте k осуществлен переход в данное состояние s^k со значением выхода z^k. Индекс J^k – вектор дискретных параметров задания, существенных при поиске множества внутренних состояний.***

III. ВОПРОСЫ ВНУТРЕННЕГО МОДЕЛИРОВАНИЯ

К внутреннему моделированию цифрового устройства относятся вопросы минимизации таблицы переходов построенного автомата, кодирования его состояний и реализации последовательностной схемы.

Предпринятое в этом разделе рассмотрение имело целью показать сложности схемной реализации (структурного синтеза) результата абстрактного синтеза автомата. В общем случае задача минимизации таблиц переходов алгоритмически неразрешима.То же относится к компактному кодированию состояний с совместным использованием строк. Не просто решить и вопросы реализуемости синхронных таблиц переходов. Так что имеет смысл задуматься о возможности представления последовательностного алфавитного оператора без этих трудностей.

IV. ФРЕЙМОВО-ПРОДУКЦИОННАЯ МОДЕЛЬ

Рост сложности задач делает необходимой автоматизацию процедуры синтеза, что требует адекватной структуризации знаний. Итогом этой структуризации является фреймово-продукционная модель. Она сводит процесс синтеза к последовательному формированию ряда таблиц-фреймов Одни из них заполняются пользователем согласно заданию. Другие представляют наборы резидентных подпрограмм. Третьи генерируются автоматически как результаты синтеза с использованием системных и присоединенных процедур.

Постулат 4.1. ***Структура фреймово-продукционной модели синтеза неформально заданных автоматов в целом отвечает показанной на рис. 4.1. Спецификация фреймов – табл. 4.1.***

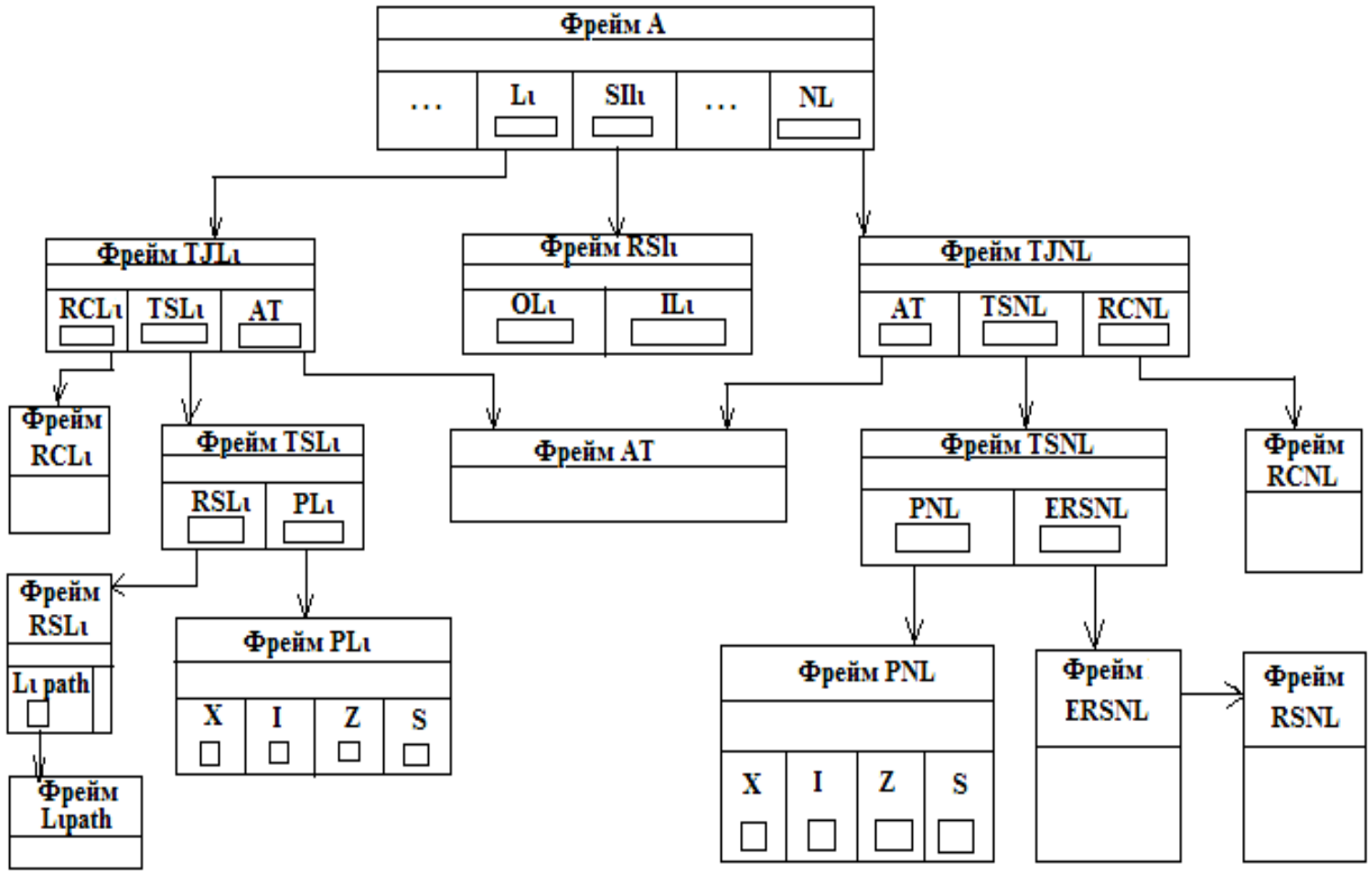

Фрейм A
...
Lı
SIIı
...
NL
Фрейм TJLı
RCLı
TSLı
AT
Фрейм RSIı
OLı
ILı
Фрейм TJNL
AT
TSNL
RCNL
Фрейм RCLı
Фрейм TSLı
RSLı
PLı
Фрейм AT
Фрейм TSNL
PNL
ERSNL
Фрейм RCNL
Фрейм RSLı
Lı path
Фрейм PLı
X
I
Z
S
Фрейм PNL
X
I
Z
S
Фрейм ERSNL
Фрейм RSNL
Фрейм Lıpath

Рис.4.1

Таблица 4.1

Имя фрейма	Семантика фрейма
A	Концептуальный целевой фрейм автомата
AT	Тип автомата
RSIℓ	Правила сопряжения областей с лабиринтом ℓ
PNL	Параметры вне лабиринтов
RSNL	Правила спецификации вне лабиринтов
TSNL	Таблица спецификации вне лабиринтов
RCNL	Правила следования вне лабиринтов
TJNL	Таблица переходов вне лабиринтов
PLℓ	Параметры в лабиринте ℓ
Lℓ path	Пути в лабиринте ℓ
RSLℓ	Правила спецификации в лабиринте ℓ
TSLℓ	Таблица спецификации в лабиринте ℓ
RCLℓ	Правили следования в лабиринте ℓ
TJLℓ	Таблица переходов в лабиринте ℓ

V. ПОСТРОЕНИЕ ИНТЕРАКТИВНОЙ СИСТЕМЫ

Предложенная фреймово-продукционная модель создает теоретические предпосылки к построению экспертных систем синтеза широкого класса автоматов. Но возникает вопрос о приемлемом инструментальном средстве для реализации построенной модели. При формировании ряда фреймов широко используется поиск в таблицах. Выполнение над таблицами операций селекции и проекции с устранением дубликатов явилось основанием для сделанного вывода о целесообразности погружения фреймовой модели в среду реляционной СУБД.

При таком погружении таблицы-фреймы представляются отношениями базы данных. Связи между фреймами организуются с помощью встроенных в СУБД средств. В качестве этой СУБД был выбран продукт фирмы Microsoft – **СУБД Access**. Особенность таблиц-фреймов состоит в том, что значения одного и того же атрибута могут принадлежать разным доменам. Значения атрибутов могут интерпретироваться и как ссылки на значения других слотов. Это обуславливает необходимость внесения определенных корректив в механизм реляционных СУБД.

Взаимосвязь фреймов осуществляется с помощью стандартных средств **Access**. В качестве этих средств выступают язык запросов **SQL** и встроенный в Access язык программирования **Visual Basic**. Связь фреймов осуществляется прямым обращением из одного фрейма к другому с использованием функций **Visual Basic** либо обращением с помощью **SQL**-запроса, когда необходимо осуществить выбор слотов другого фрейма и их обработку. Пример реализации системной процедуры генерации фрейма **TSNL** показан на рис. 5.2.

Интегрированный состав среды, включающий язык запросов, используемый для организации быстрого поиска и выбора значений в таблицах-фреймах, и язык программирования, с помощью которого организуется взаимосвязь таблиц-фреймов и обработка присоединенных процедур после их трансляции с предложенного языка, позволил построить систему, пригодную для решения задачи автоматизированного синтеза автоматов. Ее испытания прошли успешно. Результатом работы системы является неминимизированная таблица переходов автомата. Найденное решение должно быть минимизировано известными методами. Возможные ошибки в работе системы связаны с неполным учетом пользователем особенностей задания. Подсистема валидации в составе системы проверяет наличие таких ошибок после минимизации решений.

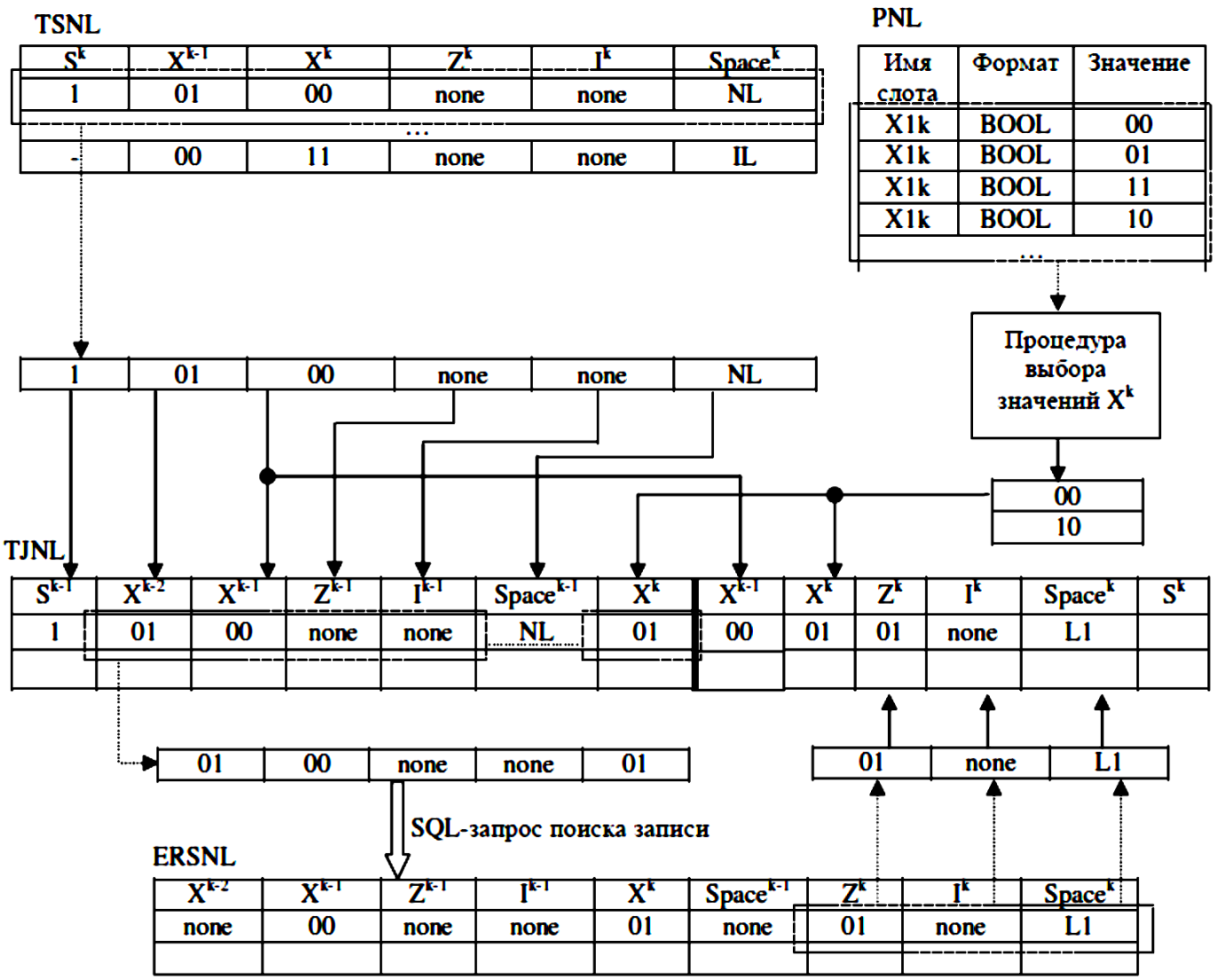

TSNL
PNL
Имя слота
Формат
Значение
Процедура выбора значений X^k
SQL-запрос поиска записи
TJNL
ERSNL

Рис. 5.2

VI. ПРОГРАММНОЕ МОДЕЛИРОВАНИЕ АВТОМАТА

Решение общей задачи синтеза автомата по неформальному заданию (включая его реализацию) с ростом ее сложности становится проблематичным. Это вызвано:

1) необходимостью валидации решений, выдаваемых интерактивной системой синтеза;

2) требованиями минимизации получаемых таблиц переходов, кодирования ее состояний (что в общем случае алгоритмически неразрешимо) и реализуемости автомата (некритичности к разбросу задержек элементов).

Хорошей альтернативой использованию автомата как такового может стать применение его программной модели при условии обеспечения необходимой скорости реакции системы.

Показывается удобство использования языка ЯРВ для написания программных моделей неформально заданных автоматов. Возможными изменениями входов/выходов автомата могут быть не только «изменение из 0 в 1 либо из 1 в 0», но и: «сохранение значения», «изменение на противоположное», «r-кратное изменение», «значения в соседних тактах различны», «определенные значения входов недопустимы» и др. Такие словесные определения можно описать одним шаблоном с помощью ЯРВ. Возможность ввода слов серьезно облегчает труд разработчика.

В этом разделе представлены два различных подхода к использованию средств **ЯРВ** для решения задачи программного моделирования процесса функционирования автомата. Основное отличие заключается в том, что первый подход предполагает неформальное (словесное) задание автомата. Во втором пользователь указывает наборы входов и выходов синтезируемого автомата и задает правила на языке **ЯРВ**, которые будут формировать базу знаний автомата. Перед применением фильтра по шаблонам создается база данных всевозможных переходов. Такая особенность позволяет ускорить процесс моделирования, т.к. для изменения модели достаточно изменить правила **ЯРВ**-фильтров. **ЯРВ** в любом случае – инструмент для достижения цели**:** разбор неформального определения автомата – в первом и фильтр возможных значений – во втором.

Первый подход ориентирован на построение подсистемы валидации интерактивной системы синтеза автоматов. В нем процесс формирования базы знаний проверяемого автомата связан с использованием инструментальной **ЯРВ**-системы и средств СУБД **Microsoft Access**, включенной в состав системы. В этом процессе при вводе пользователем каждого правила на языке, близком к естественному, генерируется фильтр, или шаблон на **ЯРВ**. Он помогает выбирать лишь необходимые правила из автоматически формируемого множества.

Искомая база знаний оформляется в виде таблицы **«Base»** базы данных **Microsoft Access.**

Есть и еще одно отличие второго подхода от предыдущего. Оно состоит в следовании принципам **КМС**. Единообразие написания **ЯРВ**-шаблонов для разнотипных автоматов достигается введением глобальных переменных, которые применяются по усмотрению пользователя. При этом не исключается расширение таблицы базы знаний атрибутами с дополнительной семантикой. В частности, – в виде вызова присоединенных процедур для вычисления значений функционально заданных параметров.

Разработанный исследовательский прототип автономной системы программного моделирования функционирования автоматов ориентирован на пользователя, который не только достаточно знаком с **ЯРВ** для написания шаблонов, но и может сделать это на основании анализа динамики автомата, согласно его описанию. Квалифицированный пользователь в состоянии написать шаблон для заданий приемлемой сложности, когда множества параметров и их значений не слишком велики, а составление шаблонов не слишком утомительно. При усложнении задания это может «вылиться» в разработку комплексных проектов силами небольшого коллектива.

Формирование базы знаний автомата в процессе валидации

Чтобы создать программную модель, надо знать совокупность правил (иметь **базу знаний**), по которым формируется последовательность значений выхода в ответ на действие входной последовательности. База знаний автомата формируется в два этапа:

***Этап 1*.** Правила функционирования автомата, введенные пользователем, переводятся на **ЯРВ**. Тем самым создается фильтр для выбора из множества всевозможных лишь необходимых правил. Правила вводятся пользователем в форме «условие–действие». Предварительно создаются таблицы базы данных **«What_Shablon»** (для условия) и **«What_Shablon_Z»** (для действия), в которых хранятся «словоформы» возможных входных и выходных изменений соответственно.

Каждое вводимое правило проходит предварительную обработку. В результате такой обработки каждого введенного правила формируется таблица шаблонов **«Filtr».** Например, при вводе пользователем правила

«Если 2-кратное изменение x_1, то z равен противоположному»

для автомата с двумя входными и одним выходным сигналом сформируется следующая строка-фильтр:

(x=((?<s1>(0|1))(0|1)-(?!\k<s1>)(0|1)(0|1)-\k<s1>(0|1)))\ s
(z=(0|1)(?<s>(0|1))-(?!\k<s>)(0|1))\s

Этап 2. На основе данных, введенных пользователем, сначала формируются массивы возможных значений входных и выходных состояний. Число таких значений равно $\mathbf{2^{число\ входов}}$ и $\mathbf{2^{число\ выходов}}$ соответственно. Затем формируется множество правил путем генерации всевозможных входных изменений различной длины, от минимальной (равной 1) до максимальной (вводит пользователь), с использованием комбинаторного алгоритма генерации размещений с повторениями и их фильтрации с использованием таблицы **«Filtr».**

«Фильтрация» и генерация происходят поэтапно. Сначала производится генерация входных изменений и их фильтрация (используется часть фильтра для условия). Затем – генерация выходных изменений и их фильтрация (используется часть фильтра для действия). Далее порождаются сочетания отфильтрованных входных и выходных изменений и происходит заключительный этап фильтрации (используется вся строка фильтра). Найденная база знаний будет представлять собой таблицу под названием **«Base»** базы данных **Microsoft Access**.

Автономная система программного моделирования автоматов

Вопросы программного моделирования функционирования автоматов были затронуты в предыдущем подразделе применительно к разработке подсистемы валидации в составе интерактивной системы с использованием ее инструментальных средств. Предметом настоящего подраздела является развитие конструктивного подхода к созданию автономной системы программного моделирования. Конструктивизм отвечает позициям **КМС**. Излагаются основные положения адаптации **ЯРВ**-подхода к моделированию цифровых автоматов, развитой на базе языка **Perl**.

Целесообразна структуризация представления правил функционирования автомата с целью дальнейшей разработки интерактивной системы моделирования. За основу структуризации принимается спецификация состояния неформально заданных автоматов кортежем $\mathbf{<(x^{k-1} – x^k), z^k, J^k>.}$

Ограничение 6.1: 1) Все события в автомате относятся к классу определенных событий. В промежутке между наступлением двух соседних событий A_{z1} и A_{z2} сохраняется значение выхода z1. 2) События длиной n > 2 разбиваются на n - 1 подсобытий длины 2. Например,

$$\mathbf{x_1x_2 = 00 – 01 – 10 \rightarrow (x_1x_2)^1 = 00 – 01,\ (x_1x_2)^2 = 01 – 10.}$$

Постулат 6.1. ***Для принятого разбиения определенных событий в случае программного моделирования функционирования автомата:***

1) Правомерна следующая продукция:

$$<(x^{k-1} – x^k), z^{k-1}, J^{k-1} > \supset < z^k, J^k >. \qquad (6.1)$$

2) Каждый параметр индекса – метка развития того или иного события в соответствующей области функционирования автомата.

Продукция 6.1 учитывает**:**

– важность в общем случае знания предыдущего значения выхода z^{k-1} для определения z^k, реализуемого отображением ***L***, в отсутствие этапа абстрактного синтеза;

– понимание под J^k вектора индекса места, занимаемого символом x^k входной последовательности на множестве определенных заданием событий.

Согласно (6.1), база знаний представляет собой таблицу, атрибуты которой суть изменения входа $(x^{k-1} – x^k)$, выхода $(z^{k-1}– z^k)$ и индекса $(J^{k-1} – J^k)$. Число параметров вектора индекса зависит от числа отрезков в событии, от количества событий и рабочих областей. Так, если событие выхода из лабиринта – единственное и содержит 2 отрезка, а область вне лабиринта не требует индексации, то надо ввести 3 параметра**:** указатели свершившихся подсобытий – **P1, P2** и указатель рабочей области («лабиринт» либо «вне лабиринта») – **P0**.

Разрядности входов и выходов автомата, число параметров вектора индекса задаются при создании автоматизированной системы. На основании этих данных формируется входной и выходной словари (одноатрибутные отношения) **words_x** и **words_z.** Например, для случая с 2 входами словарь **words_x** будет содержать 4 слова – 00, 01, 10 и 11. При задании трех параметров имеем еще 3 отношения – **(P0), (P1), (P2).** Система генерирует всевозможные события длины 2 из содержимого **words_x** и **words_z** Результатом является массив записей вида ***декартовых произведений***

$$(x^{k-1}–x^k) = (words\_x) \otimes (words\_x), (z^{k-1}–z^k) = (words\_z) \otimes (words\_z).$$

Затем формируется полноатрибутное отношение

$$(x^{k-1}–x^k) \otimes (z^{k-1}–z^k) \otimes (P0) \otimes (P1) \otimes (P2).$$

Соответствующее множество записей ***sentences*** получается весьма емким.

Процесс формирования базы знаний автомата в общем случае показан на рис. 6.1. Программа фильтрации (**FILTR**) просматривает множество записей ***sentences*** и отбирает те из них, которые удовлетворяют множеству шаблонов (***shablons***), написанных пользователем на языке **ЯРВ**. Такие записи попадают в базу знаний, описывающую искомый автомат (**BASE**).

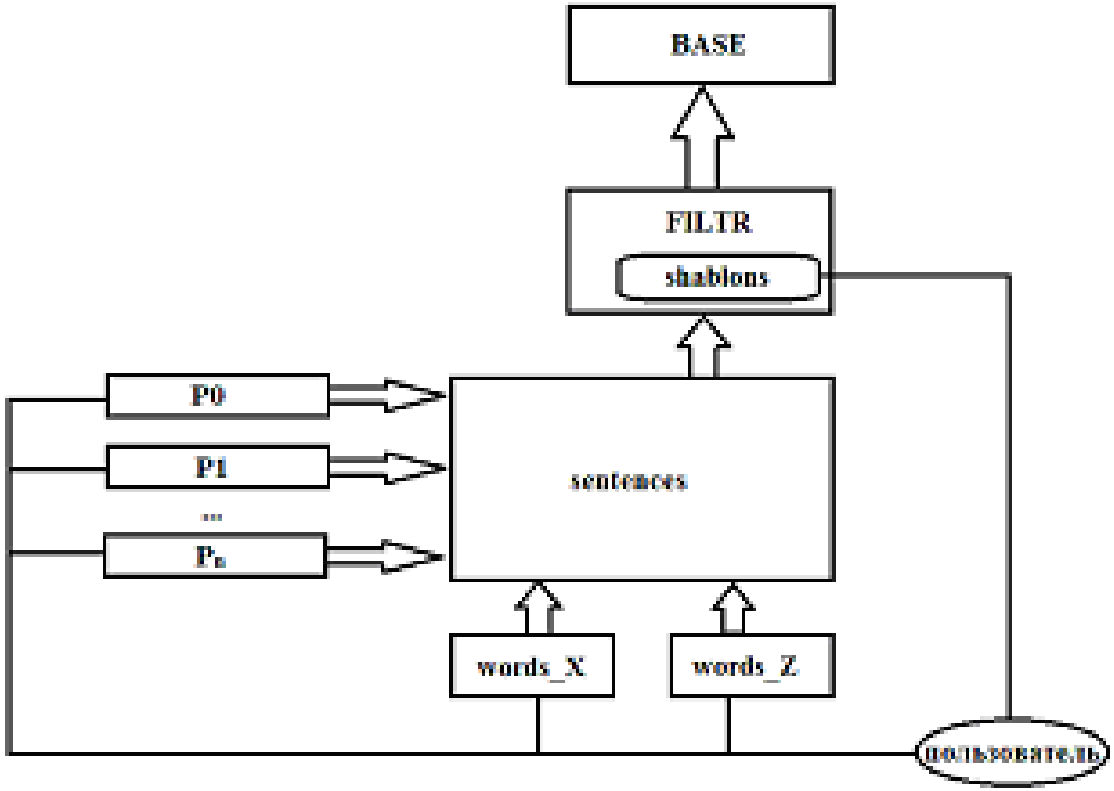

BASE
FILTR
shablons
P0
P1
...
Pn
sentences
words_X
words_Z
пользователь

Рис.6.1

Приложение I. ДИНАМИКА ЦИФРОВЫХ СХЕМ

Необходимым условием реализации цифровым устройством заданного алгоритма является правильность логического функционирования всех его блоков. Одни из них относятся к классу комбинационных схем. Другие – к классу последовательностных. Протекание переходных процессов в цифровых схемах каждого класса специфично. Учет особенностей динамики схем – важный фактор структурного проектирования по результатам логического (абстрактного) синтеза.

Синхронному принципу организации функционирования цифровых устройств (с применением сигнала синхронизации) отдается предпочтение. Но в моменты переключений любая цифровая схема ведет себя как асинхронная. Возникающий при этом риск сбоя может проявляться при некотором сочетании задержек элементов и соединений. Кроме того, использование асинхронных подходов при реализации синхронных схем может дать серьезный эффект. Поэтому все внимание в данном подразделе уделяется динамике асинхронных схем, комбинационных и последовательностных.

Приложение II. НАМЕТКИ АВТОМАТНОГО ПОДХОДА К ЗАЩИТЕ БЛА В ОПАСНЫХ СИТУАЦИЯХ

Мы полагаем возможным применение программного моделирования функционирования автоматов для целей защиты беспилотных летательных аппаратов БЛА сравнительно небольшого веса в экстремальных ситуациях (сильный боковой ветер, встречное грозовое облако, прицельно летящий снаряд и др.). Наша убежденность основана на том, что к настоящему времени накоплен немалый практический опыт логического управления реактивными системами с использованием автоматного программирования.

Такое применение подразумевает замену тяжелого труда наземного оператора функционированием специального программного модуля – Диспетчера на борту БЛА . На вход Диспетчера непрерывно поступают сигналы от всевозможных датчиков – инерциальных, оптических – о состоянии аппарата с учетом «окружения». При возникновении опасности Диспетчер выдает команды контроллерам, которые управляют работой соответствущих исполнительных механизмов – рулей поворота и высоты, двигателей.

Пока что системы автоматического управления БЛА – достаточно сложные программно-технические устройства, реализующие классические алгоритмы управления. Для БЛА, обладающих сравнительно небольшим весом, такие алгоритмы не всегда эффективны, что выдвигает на повестку дня использование для целей управления методов искусственного интеллекта с позиций экспертных систем и автоматного подхода.

Мы полагаем правомерными следующие принципы организации автоматной защиты БЛА:

– На основе анализа сигналов, поступающих от датчиков, диспетчер устанавливает наличие и вид опасности либо ее отсутствие. При наличии опасности адресуется один или несколько бортовых контроллеров, которые управляют действием механизмов, способных помочь ликвидации возникшей опасности. Код опасности подается на вход автомата в составе диспетчера. С его выхода снимается последовательность команд для инициированных контроллеров.

– Диспетчер реализует продукции:

<Сигналы от {Дi}> $\supset$ **< Вход автомата ><{Kj}>.**

Он работает с заведомо сформированной базой знаний, структурированной, как показано на рис. П2.1 (**Дi** – датчики, **Kj** – контроллеры). Уровни сигналов от датчиков на входе диспетчера: ***нормальный* – код 00, *ниже нормы* – код 10, *выше нормы* – код 01**.

Условие				***Действие***				
Коды *уровней* сигналов от датчиков				**Код опасности**	**Адреса *последовательно* инициируемых контроллеров**			
$Д_1$	**$Д_2$**	**…**	**$Д_k$**		**K_1**	**K_2**	**…**	**K_n**

Рис П2.1

– ***Управление – пошаговое.*** Для сравнительно простых ситуаций (например, сильный боковой ветер) стереотипные шаги управления, определяемые видом опасности, повторяются, пока эта опасность действует. Они прекращаются по нормализации ситуации. В более серьезных случаях (например, нахождение в грозовом облаке) организуется соответствующий маневр, в реализации которого могут принимать участие несколько механизмов. Соответственно могут быть последовательно, от такта к такту, задействованы несколько контроллеров.

– Программная модель автомата заведомо формируется для обработки мыслимого множества угроз. По нормализации ситуации автомат переводится в пассивное состояние. ***Семантика входных и выходных сигналов автомата – упрощенная: превышение порога уклонения вправо/влево → разворот влево/вправо, поражающий снаряд в том или ином направлении → соответствующий маневр и т.д.***

– ***Автомат многофункционален***, предназначен для управления рядом исполнительных механизмов в различных опасных ситуациях. В табл. П2.1 представлен синхронный автомат для реализации двух способов защиты: лево/право и маневр.

Таблица П2.1

y^{k-1} \ X^k	X_o:=00	$X_{лев}$:=01	$X_{пр}$:=11	$X_м$:=10
1	1, z_o	2, $z_{лев}$	2, $z_{пр}$	2, z_1
2	1, z_o	3, $z_{лев}$	3, $z_{пр}$	3, z_2
3	1, z_o	4, $z_{лев}$	4, $z_{пр}$	4, z_3
4	1, z_o	5, $z_{лев}$	5, $z_{пр}$	5, z_4
5	1, z_o	1, $z_{лев}$	1, $z_{пр}$	1, z_5

Здесь $X^k \neq X_o$ – код опасности. Коды z_i, $i \in \{1, 2, \ldots\}$ – бинарные коды команд для последовательно инициируемых контроллеров. Значение $X^k = X_o$ говорит об отсутствии опасности. Если ситуация не нормализуется на пятом шаге (наш пример), то процесс повторяется либо, в случае маневра, переходит на иную правильную последовательность. По ликвидации опасности выполняется переход в столбец $X^k = X_o$. Автомат и контроллеры переводятся в ждущее состояние. Управление передается автопилоту.

Реализация такого подхода подразумевает участие экспертов – специалистов по пилотажу и двигателям, по управлению БЛА. Они выявляют связь порядка запуска того или иного механизма с той или иной внештатной ситуацией, заполняя тем самым базу знаний рис. П2.1.

Вопрос практической реализации Диспетчера непосредственно на «борту» аппарата требует специального рассмотрения. ***Не исключена целесообразность организации его «наземного» функционирования.*** Тогда система защиты в целом приобретает черты ***коллаборативного*** (от англ. ***collaboration*** – коллективная деятельность, сотрудничество) ***распределенного в пространстве робота***. Его «глаза» (датчики) и исполнительные механизмы (рули, двигатели) находятся на борту БЛА. Информация от датчиков поступает по радиоканалу на землю и обрабатывается в реальном времени на достаточно мощном сервере. Результаты обработки в виде последовательности двоичных команд передаются по тому же каналу на исполнительные механизмы.