ISSN 0136-3549
3134-3823

TALLINNA
POLÜTEHNILISE INSTITUUDI
TOIMETISED

TPI '83

550

ТРУДЫ ТАЛЛИНСКОГО
ПОЛИТЕХНИЧЕСКОГО
ИНСТИТУТА

РАСЧЕТ И ПРОЕКТИРОВАНИЕ СИСТЕМ ТЕХНИЧЕСКОЙ КИБЕРНЕТИКИ

550

TPI '83

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED

ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

УДК 681.5+681.32

РАСЧЕТ И ПРОЕКТИРОВАНИЕ СИСТЕМ ТЕХНИЧЕСКОЙ КИБЕРНЕТИКИ

Электротехника и автоматика XXII

Таллин 1983

Teaduslik Raamatukogu
ENSV TA
TALLINN
АН ЭССР
Научная библиотека

© Таллинский политехнический институт, 1983

№ 550

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED

ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

УДК 658.52.011.56

О.А. Аарна

МАТЕМАТИЧЕСКАЯ МОДЕЛЬ НЕПРЕРЫВНОГО АВТОМАТИЗИРОВАННОГО ТЕХНОЛОГИЧЕСКОГО КОМПЛЕКСА И ЕЕ ПРИМЕНЕНИЕ В АСУ ТП

I. Введение

Расширение фронта работ по созданию и внедрению АСУ ТП привело к появлению нового обширного класса человеко-машинных систем, называемых автоматизированными технологическими комплексами (АТК). Под непрерывным АТК будем понимать совокупность непрерывного технологического объекта управления и его АСУ [I]. Основными компонентами АТК и АСУ ТП являются:

1) техническое обеспечение,
2) программное обеспечение,
3) информационное обеспечение,
4) оперативный персонал.

Каждый из перечисленных компонентов, в свою очередь, состоит из множества элементов, причем понятие элемента не фиксировано, а зависит от цели исследования. Процесс функционирования АТК сводится к сложному информационному взаимодействию этих элементов.

Для целенаправленного построения АСУ ТП необходимо уметь промоделировать работу АТК. При этом, как всегда в процессе моделирования, мы абстрагируемся от многих второстепенных характеристик объекта, выделяя существенное. В настоящей работе этими существенными характеристиками АТК приняты информационная связанность его элементов и наличие у каждого элемента конечного набора состояний. Из последнего следует, что мы рассматриваем АТК как специфическую динамическую систему, а любая задача управления в такой

системе естественным образом распадается на две дополняющие друг друга подзадачи:

I) оценки состояния по наблюдаемым выходам объекта,

2) выработки управляющих воздействий на объект в функции от полученных оценок состояния.

Показано, как эта общая концепция состояния применяется при синтезе АТК.

2. Модель непрерывного АТК

Структура АТК описывается направленным графом $\Gamma=(T,K)$, вершинами T которого являются элементы технического обеспечения T_t программного обеспечения T_p информационного обеспечения T_i и оперативного персонала T_0: $T=T_t \cup T_p \cup T_i \cup T_0$ ($T_k \cap T_l \neq 0$ для всех $k,l=t,p,i,0$ и $k \neq l$), а дуги K соответствуют передаче информации между названными элементами, безотносительно ее материального носителя. Назовем Γ информационным потоковым графом (ИПГ) АТК. В качестве подграфов ИПГ можно выделить граф технической структуры, программного обеспечения и граф структуры информационного обеспечения АТК. Эти подграфы порождены соответственно множествами вершин T_t, T_p и T_i. ИПГ АТК и некоторые его подграфы являются связными.

С каждой вершиной $t_j \in T$ связано конечное множество состояний Z_j рассматриваемого элемента. Состояние АТК Z является элементом прямого произведения $z \in Z = Z_1 \times Z_2 \times Z_3 \times \ldots \times Z_{n_T}$, где n_T - число вершин ИПГ. Множество допустимых состояний АТК Ω является подмножеством $z: \Omega \subset Z$.

Особенностью непрерывных АТК является то, что с некоторыми вершинами $t_k \in T_k$ (технологические аппараты, их части или комплексы, технологические потоки) связаны непрерывные векторные пространства состояний. Для каждой такой вершины дополнительно установлено соответствие между ее дискретными состояниями и замкнутыми или открытыми областями непрерывного пространства состояний. Поскольку чаще всего эти области оказываются пересекающимися, то названное соответствие будет размытым. Следует подчеркнуть второстепенность непрерывного пространства состояний с точки зрения моделирования АТК.

Процесс функционирования АТК сводится к смене состояний его элементов. Переход из одного состояния в другое происходит спонтанно или по соответствующей команде(входному символу). Примером спонтанного перехода может служить выход из строя некоторого элемента. Имеются недопустимые переходы состояний элементов. Учитывая конечность множества состояний элементов, удобно описывать переходы состояний с помощью принятого в теории конечных автоматов графа переходов состояний [2]. В этом графе только детерминированным переходам сопоставлены входные символы. Спонтанные переходы состояний элементов можно обнаружить на основе наблюдаемых выходов АТК, но их нельзя вызвать по команде.

Таким образом, моделью АТК является семейство взаимосвязанных конечных автоматов с частично детерминированными переходами состояний. Это означает, что модель АТК в целом также представляет собой конечный автомат, причем сложность ее поведения обусловлена сложностью информационных связей между элементами АТК. Особенностью нашей автоматной модели АТК является отсутствие в ней времени в явном виде.

3. Анализ модели АТК

Для ИПГ определена операция конденсации некоторого числа смежных вершин в одну агрегированную вершину. Эта новая вершина также обладает своим множеством состояний Z_a, причем чаще всего конденсация сопровождается сокращением числа состояний: $|Z_a| < |Z_{1a} \times Z_{2a} \times \ldots \times Z_{na}|$, где Z_{ja}, $j = 1, \ldots, n_a$ – множества состояний конденсированных вершин исходного ИПГ. Таким образом, после конденсации уже не все состояния элементов образованного агрегата различимы.

Имеется значительное количество конденсированных ИПГ, образуемых из одного прототипа. Эти конденсированные ИПГ соответствуют различным точкам зрения разработчика или пользователя на АТК и отличаются друг от друга как степенью детализации описания, так и способом образования агрегированных вершин.

В числе агрегированных вершин ИПГ можно выделить пары "объект управления (ОУ) – управляющее устройство (УУ)",

имеющие двусторонние информационные связи. Каждую такую пару можно рассматривать как подкомплекс АТК. Всю остальную часть ИМГ мы интерпретируем относительно рассматриваемой пары как окружающую среду. В ИПГ АТК имеется по меньшей мере одна пара "ОУ - УУ", для которой окружающая среда является пустым множеством. Такое УУ назовем системным монитором.

Наряду с операцией конденсации смежных вершин ИПГ имеется обратная операция расщепления вершины с раскрытием ее внутренней структуры. Расщепление вершины ИПГ может сопровождаться расширением соответствующего множества состояний.

Основной задачей анализа модели АТК является идентификация его состояния. Информация о состоянии АТК содержится в наблюдаемых выходах его элементов. Такими наблюдаемыми выходами являются сигналы дискретных датчиков и сообщения, вырабатываемые программами (задачами) АСУ ТП. Задача идентификации состояния АТК и отдельных его частей относится к области распознавания образов. Поскольку идентификация состояния происходит в автоматизированной системе, то она может осуществляться как автоматически, так и с привлечением человека - оператора. Соответствующее решение о способе идентификации состояния принимается при построении модели АТК.

4. Синтез модели АТК

Синтез модели АТК моделирует сам процесс построения АСУ ТП. При синтезе модели АТК заданы структура и параметры неизменяемой части комплекса (технологический объект управления вместе с локальной автоматикой и устройствами связи с объектом (УСО) или без них), а также перечень выполняемых АСУ ТП информационных и управляющих функций вместе с требованиями к качеству и надежности их выполнения.

Процесс синтеза модели АТК можно описать в виде следующего алгоритма:

I. Для каждой выполняемой АСУ ТП функции вводим одну агрегированную вершину УУ и определим для него множество вершин заданной части ИПГ, образующее ОУ для данного УУ, а также установим информационные связи между ними. В случае информационной функции связь ОУ и УУ может быть одно-

сторонней. Таким образом множественность ОУ в модели АТК является следствием множественности выполняемых АСУ ТП функций.

2. Для каждой вершины ОУ определим множество ее состояний, входной и выходной алфавит, а также граф переходов состояний. Другими словами, каждый элемент неизменяемой части АТК опишем как конечный автомат.

3. Определим возможные способы агрегации элементов ОУ с точки зрения выполняемой функции.

4. Для каждой функции АСУ ТП определим перечень действий, необходимых для ее выполнения и их информационную взаимосвязь.

5. Проанализируем зависимость выполнения этих действий от состояния объекта. Все состояния объекта, в которых выполняются одни и те же действия, неразличимы с точки зрения УУ и мы агрегируем их в одно состояние. Если при анализе выяснится, что выполнение некоторых действий зависит от прошлых состояний объекта, то число состояний ОУ необходимо увеличить.

6. Для каждого состояния ОУ определим способ его идентификации по наблюдаемым выходам объекта (автоматический или с привлечением человека-оператора).

7. Объединим выполняемые в одном состоянии объекта действия в одну задачу и определим выходные символы (сообщения) каждой задачи. Действия, повторяющиеся в нескольких задачах, целесообразно оформить в виде самостоятельных программных модулей.

8. Расщепим вершину УУ, вводя вместо него вершины задач, включая задачу монитора рассматриваемой функции. Последний состоит из идентификатора состояния объекта и исполняющей части, управляющей работой остальных задач.

9. Определим число системных мониторов и ОУ для каждого из них. Далее определим состояния объекта с точки зрения монитора, способ их идентификации, объем выполнения информационных, управляющих и вспомогательных функций АСУ ТП, а также действия монитора в каждом из рассматриваемых состояний. Системный монитор является расширением операцион-

ной системы реального времени, в котором воплощена специфика технологического объекта управления и выполняемых АСУ ТП функций.

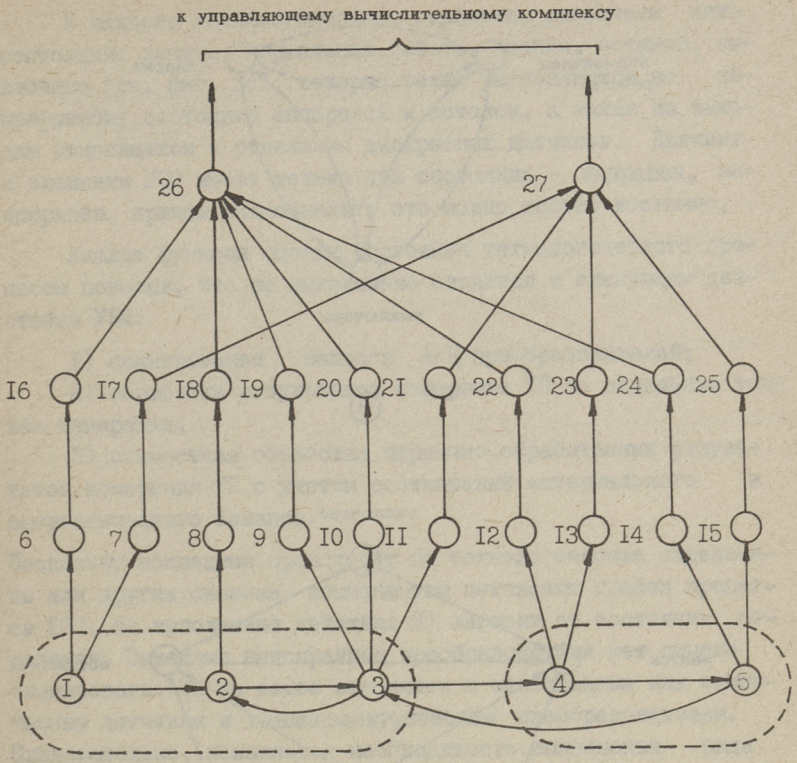
Как любой процесс построения сложной системы, синтез модели АТК является итеративным. Новую итерацию можем начать с ввода изменений в заданную часть АТК (изменение технологической системы, добавление точек измерения и т.п.) или с уточнения, расширения или сужения выполняемых функций, изменения требований к надежности и т.д. Важно то, что сохраняется общий методологический принцип, основанный на концепции состояния: каждой выполняемой функции соответствует свой ОУ, состояние которого необходимо знать с тем, чтобы целенаправленно им управлять. Если несколько функций АСУ ТП имеют близкий или совпадающий ОУ, то они могут иметь общий монитор.

5. Пример

Проиллюстрируем применение алгоритма синтеза модели АТК на примере функции оценки состояния химико-технологического процесса. Неизменяемая часть АТК состоит из технологических аппаратов, связанных материальными и энергетическими потоками в технологическую схему, датчиков температуры, давления, расхода и других технологических параметров (ТП), датчиков положения клапанов, магнитных пускателей насосов, пневмоэлектрических и аналого-цифровых (А/Ц) преобразователей (см. фиг. I). Все перечисленные элементы образуют с точки зрения функции оценки состояния ОУ, от состояния которого зависит выполнимость и способ выполнения данной функции.

По функциональному признаку технологические аппараты объединены в технологические отделения (на фиг. I обведены пунктиром). Все датчики и элементы УСО можно сгруппировать по технологическим аппаратам. Кроме того, А/Ц преобразователи конструктивно сгруппированы по монтажным платам и т.д. В результате мы получим целый ряд различных структурных моделей ОУ.

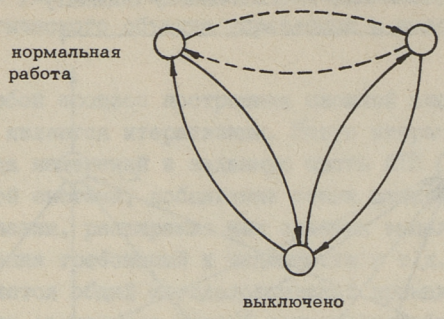
Вводим для каждого элемента неизменяемой части АТК и образованных из них агрегатов множества состояний. Каждый



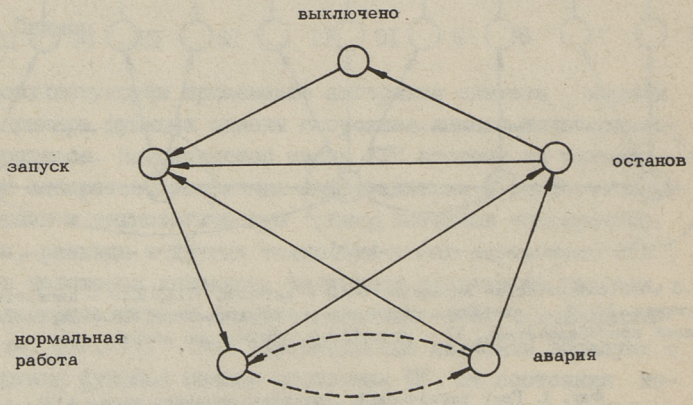
1-5 - технологические аппараты, 6-15 - датчики, 7,11,13,15 - дискретные датчики, 16-25 - пневмоэлектрические преобразователи, 26 - аналого-цифровой преобразователь, 27 - устройство дискретного ввода.

Фиг. 1. Граф технической структуры заданной части АТК.

технологический аппарат имеет три состояния: выключен, нормальная работа, авария (см. фиг. 2а). Аварийное состояние аппарата распознается по его непрерывному состоянию, т.е. по обработанным показаниям аналоговых датчиков. Переход из состояния "выключено" в состояние нормальной работы происходит по детерминированному входному сигналу (например, включение некоторого насоса). Переход в аварийное состояние и обратно в режим нормальной работы происходит спонтанно.



а



б

– детерминированные переходы

– спонтанные переходы

а – технологический аппарат

б – технологическое отделение

Фиг. 2. Графы перехода состояний.

У каждого технологического отделения различаем пять состояний: запуск, нормальная работа, авария, останов, выключено (см. фиг. 2б), которые также распознаются по непрерывному состоянию аппаратов и потоков, а также по выходам относящихся к отделению дискретных датчиков. Датчики и элементы УСО имеют только два состояния - исправен, неисправен, причем распознавать это можно только косвенно.

Анализ функции оценки состояния технологического процесса показал, что ее выполнение сводится к следующим действиям УВМ:

1) сканирование выходов А/Ц преобразователей;

2) обработка результатов измерения ТП по отдельным точкам измерения;

3) совместная обработка первично обработанных результатов измерения ТП с учетом соотношений материального и энергетического баланса.

Поскольку последнее происходит по технологическим отделениям или другим связным компонентам потоковых графов процесса [3], то выполнение действия 3) зависит от состояния отделения. Заведомо неисправные преобразователи нет смысла сканировать. То же самое относится к неисправным или отключенным датчикам и пневмоэлектрическим преобразователям. Следовательно, появляется необходимость выполнения ряда вспомогательных действий:

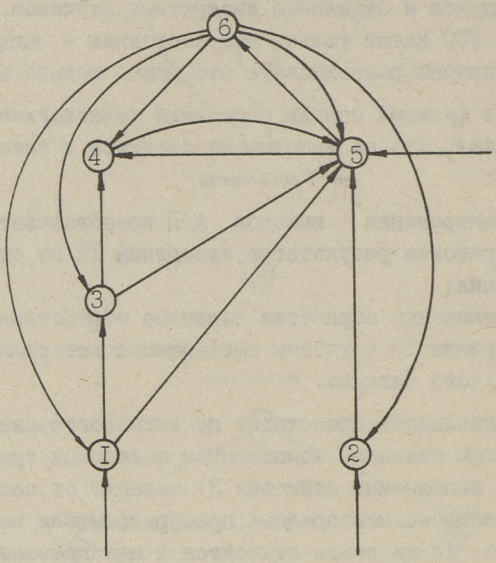
4) идентификация состояния технологических аппаратов и отделений;

5) идентификация состояния датчиков;

6) идентификация состояния УСО.

Действия 4) - 6) можно объединить общим названием - идентификация состояния ОУ. При этом очевидно, что объем и способ выполнения действий 1) - 3) зависят от состояния объекта.

При более детальном анализе основных действий выясняется еще целый ряд вспомогательных действий, выполнение которых зависит от состояния ОУ и АСУ ТП в целом. На фиг. 3 приведен граф структуры программного обеспечения функции оценки состояния технологического процесса. Относительная немногочисленность задач обусловлена тем, что каждая из



1 – спанирование А/Ц преобразователей,
2 – спанирование дискретных датчиков,
3 – первичная обработка результатов измерения ТП,
4 – оценка состояния аппаратов и отделений на базе моделей материального и энергетического баланса,
5 – идентификация технологической структуры процесса и системы технологических измерений,
6 – монитор функции оценки состояния технологического процесса.

Фиг. 3. Структура программного обеспечения функции оценки состояния технологического процесса.

них, зависящая от состояния ОУ, настраивается перед запуском на текущее состояние объекта. Информация о состоянии рассматриваемого ОУ передается в агрегированном виде системному монитору для идентификации состояния АТК в целом.

Литература

I. Общеотраслевые руководящие методические материалы по созданию АСУ ТП. Государственный комитет СССР по науке и технике. М., Финансы и статистика, 1982. 128 с.

2. Гилл А. Введение в теорию конечных автоматов. М., Наука, 1966. 272 с.

3. Аарна О. Декомпозиция задачи оценки состояния химико-технологической системы. См. наст. сб., с. 15.

O. Aarna

MATHEMATICAL MODEL OF CONTINUOUS COMPUTERIZED CONTROL PLANT

Summary

Mathematical model of continuous computerized control plant comprising continuous process plant and its computer control system is proposed. The model represents the plant hardware, software and operating personnel elements as an interconnected set of finite automata with partially determined state transitions.

№ 550

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED

ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

УДК 66.01.1.001.57(75)

О.А. Аарна

ДЕКОМПОЗИЦИЯ ЗАДАЧИ ОЦЕНКИ СОСТОЯНИЯ ХИМИКО-ТЕХНОЛОГИЧЕСКОЙ СИСТЕМЫ

I. Введение

В работе [I] для математического описания химико-технологической системы (ХТС) были введены динамические и квазистационарные балансовые модели, связывающие состояния системы с измеряемыми технологическими параметрами (ТП): температурами, давлениями, расходами, уровнями, концентрациями и т.п. В понятие состояния ХТС входят накопленные в подсистемах количества вещества, энергии (тепла) и компонентов, а также потоки этих величин между подсистемами. По своей структуре эти балансовые модели подходят для оценки состояния ХТС с помощью фильтра Калмана (ФК) [2].

Оценка состояния ХТС основана на применении временной и пространственной избыточности информации, содержащейся в результатах измерения ТП, а также в уравнениях материального и энергетического баланса, связывающих координаты оцениваемого вектора состояния и в структуре системы технологических измерений. Так как объем вычислений в алгоритме ФК пропорционален кубу размерности оцениваемого вектора состояния, а объем памяти ЭВМ пропорционален квадрату этой величины, следует искать пути для декомпозиции этой задачи [3]. В настоящей работе показано, как при этом использовать структурные и функциональные особенности применяемых балансовых моделей.

2. Дискретные балансовые модели ХТС и оценка состояния

Дискретную динамическую балансовую модель ХТС можно представить в виде стохастической системы:

$$\tilde{x}(k+1) = F(k)\,\tilde{x}(k) + \tilde{v}(k)$$
$$y(k) = C(k)\,\tilde{x}(k) + w(k), \qquad (I)$$

где $\tilde{x}=(x, q)$ – вектор состояния ХТС,
y – вектор измеряемых ТП,
$\tilde{v}$ и w – случайные последовательности с известными математическими ожиданиями и матрицами ковариации.

Матрицы $F(k)$ и $C(k)$ имеют специфическую блочную структуру:

$$F(k) = \begin{pmatrix} I_x & \tau_k A \\ 0 & I_q \end{pmatrix}, \quad C(k) = \begin{pmatrix} C_x(k) & 0 \\ 0 & C_q(k) \end{pmatrix},$$

где I_x и I_q – единичные матрицы размерности n_x (число подсистем) и n_q (число потоков),
τ_k – длительность k-го интервала дискретности,
A – блочно-диагональная матрица инциденций потокового мультиграфа ХТС Γ,
C_x и C_q – матрицы, каждая строка которых содержит не более двух ненулевых элементов.

В квазистационарной модели ХТС вектор состояния $\tilde{x}$ не содержит вектора аккумуляции x, а вектор потоков q разбит на два блока: $q=(q^t, q^i)$, где q^i – вектор независимых потоков, а q^t – вектор зависимых потоков, причем

$$q^t = A_i q^i, \qquad (2)$$

где $A_i = -A_1^{-1} A_2$, а разбиение матрицы на блоки $A=(A_1, A_2)$ соответствует принятому разбиению потоков. Матрицы $F(k)$ и $C(k)$ имеют следующий вид:

$$F(k) = I_{q^i}, \quad C(k) = C_{q2}(k) - C_{q1}(k) A_i,$$

где матрица C_q также разбита на блоки $C_q=(C_{q1}, C_{q2})$ согласно разбиению вектора потоков.

Каждая строка матрицы $(C_{q2}-C_{q1}A_i)$ может содержать произвольное число потоков ненулевых элементов.

Оценка состояния дискретной системы (I) с помощью ФК сводится к рекурсивному выполнению алгоритма:

$$\tilde{\hat{x}}(k+1) = F(k)\,\hat{\tilde{x}}(k) + K(k+1)\,\delta(k)$$
$$\delta(k+1) = y(k+1) - C(k+1)\,F(k)\,\hat{\tilde{x}}(k) \qquad (3)$$

где $\hat{\tilde{x}}$ - оценка вектора состояния ХТС,

$K(k+1)$ - матрица коэффициентов усиления ФК, вычисляемая независимо от результатов измерения ТП на основе ковариационных матриц случайных последовательностей $\tilde{v}(k)$ и $w(k)$.

Сокращенно алгоритм (3) можно представить в виде:

$$\hat{\tilde{x}}(k+1) = K(k+1)\, y(k+1) + h(\hat{\tilde{x}}), \qquad (4)$$

подчеркивающем линейность получаемых оценок относительно результатов измерения ТП.

3. Декомпонируемость задачи оценки состояния

Назовем задачу оценки состояния ХТС с помощью ФК декомпонируемой, если найдется такое разбиение векторов $\hat{\tilde{x}}$ и y из (4) на блоки: $\hat{\tilde{x}} = (\hat{\tilde{x}}^1, \hat{\tilde{x}}^2, \ldots, \hat{\tilde{x}}^m)$, $y = (y^1, y^2, \ldots, y^m)$, что для каждого $i = 1, \ldots, m$ справедливо

$$\hat{\tilde{x}}^i(k+1) = K_i(k+1)\, y^i(k+1) + h^i(p), \qquad (5)$$

где p - вектор параметров, а $m > 1$.

Легко показать, что для декомпонируемости задачи оценки состояния необходимо и достаточно, чтобы матрица коэффициентов усиления ФК $K(k+1)$ была приводима к блочно-диагональному виду:

$$K = \mathrm{diag}(K_1, K_2, \ldots, K_m). \qquad (6)$$

Из анализа алгоритма ФК следует, что матрица $K(k+1)$ приводима к виду (6) только, если для всех $k = 0,1,\ldots$ блочно-диагональными являются матрицы $P(0)$, $F(k)$, $G(k)$, $V(k)$ и $W(k)$, где $P(0)$ - матрица ковариации оценок $\hat{\tilde{x}}(0)$, а $V(k)$ и $W(k)$ - матрицы ковариации случайных векторов $\tilde{v}(k)$ и $w(k)$. Дополнительно необходимо, чтобы разбиения перечисленных матриц на блоки соответствовали одному и тому же разбиению векторов $\hat{\tilde{x}}$ и y.

Декомпозиция задачи оценок состояния ХТС основана на том, что вектор h^i в уравнении (5) может зависеть от параметров, причем этими параметрами могут быть:

1) оценки вектора состояния $\hat{\tilde{x}}^j$ из других подзадач $j \neq i$,

2) неоцениваемые потоки $\bar{q}^i$.

$$h^i = h^i(\hat{\bar{x}}^j, \bar{q}^i). \qquad (7)$$

Матрицу перехода состояний $F(k)$ можем привести к блочно-диагональной форме, учитывая специфику структуры матрицы, если потоковый граф системы состоит из m компонентов связности. Такими компонентами связности являются, например, материальный потоковый граф, тепловой потоковый граф и потоковые графы компонентов. Этот тип декомпозиции на подзадачи оценки состояния разноименных балансовых подмоделей назовем функциональным. Имеется также возможность увеличить число компонентов связности потокового графа ХТС путем перевода некоторых потоков из числа оцениваемых в неоцениваемые. Такой тип декомпозиции задачи оценки состояния назовем структурной. В общем случае функциональную и структурную декомпозицию можно применять комбинированно, поскольку они не взаимоисключающие.

4. Функциональная декомпозиция

В работе [4] было показано, что совместность балансовых подмоделей проявляется в том, что уравнения выхода подмоделей теплового и покомпонентного материального баланса содержат компоненты вектора состояния подмодели общего материального баланса $\tilde{x}^m$. Если первой решать задачу оценки состояния подмодели общего материального баланса $\hat{\tilde{x}}^m$, то в подмоделях теплового и покомпонентного материального баланса эти оценки уже будут параметрами:

$$h^i = h^i(\hat{\tilde{x}}^m). \qquad (8)$$

Тем самым уравнения выхода подмодели покомпонентного материального баланса становятся линейными. Нелинейность сохраняется только в уравнениях выхода подмодели теплового баланса ХТС.

Качественный анализ совместных балансовых моделей показал возможность увеличения числа оцениваемых материальных потоков [5]. Однако в численных экспериментах была выявлена низкая точность таких оценок. Точность оценок совместной модели только на несколько процентов выше точности оценок, полученных из функционально декомпонированной

модели. Это позволяет рекомендовать и в любом случае использовать функциональную декомпозицию задачи оценки состояния ХТС.

5. Структурная декомпозиция

Структурная декомпозиция задачи оценки состояния ХТС основана на том, что каждой подзадаче соответствует связной компонент (или несколько компонентов связности) потокового графа системы, а увеличить число таких компонентов можно, переводя некоторые потоки из числа оцениваемых в неоцениваемые. Тем самым вектор h^i из уравнения (5) будет функцией неоцениваемых потоков $\bar{q}_y^i$, инцидентных рассматриваемому i-му связному компоненту:

$$h^i = h^i(\bar{q}_y^i). \qquad (9)$$

Все m подзадач оценки состояния с вектором h^i типа (9) полностью независимы и могут быть решены параллельно, например, на микро-ЭВМ.

С учетом наличия неоцениваемых потоков уравнения состояния динамической балансовой модели (I) приобретают вид:

$$\begin{aligned} x(k+1) &= x(k) + \tau_к [\bar{A} q_y(k) + B \bar{q}_y(k)] + v^x(k) \\ q_y(k+1) &= q_y(k) + v^{q_y}(k), \end{aligned} \qquad (10)$$

где $A = (\bar{A}, B)$, а вектор $v^x(k)$ моделирует неопределенность, вносимую неоцениваемыми потоками.

Некоторые потоки могут играть в задачах оценки состояния двоякую роль: быть в одной подзадаче оцениваемыми, а во всех остальных неоцениваемыми. Тем самым устанавливается частичный порядок решения подзадач оценки состояния ХТС. Частичное упорядочение подзадач возможно только в случае, если конденсированный потоковый граф, в котором вершинами служат связные компоненты Γ_i, а дугами - дуальные потоки, не содержит циклов. Для разрыва циклов конденсированного потокового графа подходящим образом выбранные потоки должны быть объявлены неоцениваемыми, но не дуальными.

Число компонентов связности потокового графа ХТС ограничено только числом его вершин. Однако искусственное увеличение числа компонентов связности приводит к уменьшению

пространственной избыточности информации, используемой при образовании оценок вектора состояния. В предельном случае независимой оценки каждой координаты этого вектора остается лишь временная избыточность информации. При наличии возможности уменьшения длительности такта τ_k доля временной избыточности информации в повышении точности оценок состояния значительно выше доли пространственной избыточности. В каждом конкретном случае выбор степени структурной декомпозиции потокового графа системы является вопросом рационального компромисса между объемом вычислений и точностью получаемых оценок.

6. Декомпозиция квазистационарной модели

Декомпозиция задачи оценки состояния квазистационарной модели ХТС

$$\begin{aligned} q_{\gamma}^{i}(k+1) &= q_{\gamma i}(k) + v^{i}(k) \\ y(k) &= [C_{q2}(k) - C_{q1}(k)\,A_i]\,q_{\gamma}^{i}(k) + w(k) \\ q_{\gamma}^{t} &= A_i\,q_{\gamma}^{i} \end{aligned} \qquad (II)$$

заслуживает отдельного рассмотрения ввиду особенностей ее структуры и приводимости всех динамических балансовых моделей (I) к квазистационарным. Последнее происходит путем расширения вектора потоков исходной модели $q_{\gamma}(k)$ потоками аккумуляции $q_{\gamma}^{a}(k) = x(k+1) - x(k)$. В потоковом графе системы такому преобразованию соответствует переход к циклическому потоковому графу (ЦПГ) путем добавления новых вершин x_{0l} согласно числу разноименных балансовых подмоделей и заходящих в эти вершины потоков аккумуляции. Зависимые потоки образуют каркас полученного ЦПГ [I].

Поскольку в квазистационарной модели матрица перехода состояний единичная, то декомпонируемость задачи оценки ее состояния зависит полностью от структуры матрицы $[C_{q2}(k) - C_{q1}(k)\,A_i]$. Ниже предположено, что функциональная декомпозиция произведена и каждая строка матрицы $C_{q}(k)$ содержит ровно один ненулевой элемент.

Если все n_{yq} измеряемых потоков относятся к независимым, т.е. являются хордами ЦПГ при данном каркасе, то задача оценки состояния модели (II) распадается на n_{yq}

независимо (параллельно) решаемых одномерных задач оценки состояния моделей типа

$$q_j^i(k+1) = q_j^i(k) + v_j^i(k)$$
$$y_j(k) = C_j(k)\, q_j^i(k) + w_j(k) \qquad (I2)$$

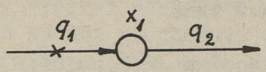
$j = 1, \ldots, n_{yq}$ и последующее вычисление оценок зависимых потоков по формуле (2). Если же среди измеряемых потоков появляются зависимые, то часть или все независимые потоки необходимо оценивать совместно. Для нахождения совместно оцениваемых независимых потоков также можно воспользоваться ЦПГ системы, учитывая, что строки матрицы A_i суть простые разрезы потоковых графов, полученных агрегацией вершин ЦПГ, связанных зависимыми потоками, если поочередно разрывать измеряемые потоки.

Проиллюстрируем сказанное на простом примере ХТС с потоковым графом, приведенном на фиг. I а (крестики обозначают измеряемые потоки, аккумуляция в подсистеме x_1 измеряется.) ЦПГ соответствующей квазистационарной модели показан на фиг. I б. Каркас ЦПГ содержит одну дугу.

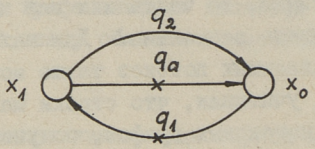
Если зависимым выбрать неизмеряемый поток $q_{\gamma 2}$, то есть $q_\gamma^i = (q_{\gamma 1}, q_{\gamma a})$, то при разрыве потоков $q_{\gamma 1}$ или $q_{\gamma a}$ ЦПГ агрегируется в одну вершину. Это является общим свойством всех измеряемых независимых потоков и соответствует уравнениям типа (I2). Если же зависимым выбрать измеряемый поток $q_{\gamma 1}$, т.е. $q_\gamma^i = (q_{\gamma 2}, q_{\gamma a})$, то при разрыве потока $q_{\gamma 1}$ получается граф, приведенный на фиг. I в. Простым разрезом этого графа является $\{q_{\gamma 2}, q_{\gamma a}\}$. Это означает, что $q_{\gamma 2}$ и $q_{\gamma a}$ необходимо оценивать из совместной модели:

$$q_{\gamma 2}(k+1) = q_{\gamma 2}(k) + v_2(k)$$
$$q_{\gamma a}(k+1) = q_{\gamma a}(k) + v_a(k)$$
$$y_{q_{\gamma a}}(k) = C_{q_{\gamma a}}(k)\, q_{\gamma a}(k) + w_a(k) \qquad (I3)$$
$$y_{q_{\gamma 1}}(k) = C_{q_{\gamma 1}}(k)\,[q_{\gamma 2}(k) + q_{\gamma a}(k)] + w_{q_{\gamma 1}}(k).$$

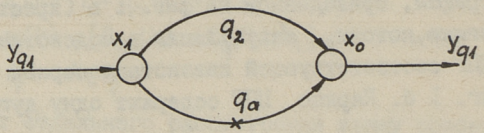
Таким образом, в зависимости от выбора независимых потоков, приходим к качественно различным задачам оценки состояния. Подходящим выбором независимых потоков можно управлять степенью декомпозиции задачи оценки состояния квазистационар-



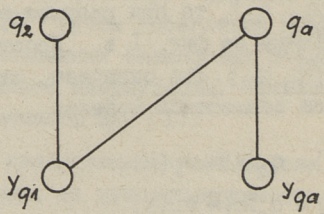
а



б



в



г

а – потоковый граф динамической балансовой модели,
б – ЦПГ квазистационарной модели,
в – ЦПГ после разрыва потока q_1,
г – двудольный граф структуры уравнений выхода модели (13).

Фиг. 1.

ной модели. Возможен также перевод некоторых потоков из числа оцениваемых в неоцениваемые.

При анализе декомпонируемости более сложных квазистационарных моделей полезным оказывается построение двудольных графов типа, приведенного на фиг. I г. Эти графы описывают структуру матрицы $[C_{q2} - C_{q1} A_i]$ и их связные компоненты соответствуют совместно оцениваемым потокам.

7. Многоуровневые схемы оценки состояния

Идея многоуровневого подхода к задаче оценки состояния ХТС заключается в агрегации группы подсистем, составляющих связной компонент потокового графа Γ_i, в одну подсистему. При этом, естественно, происходит потеря части информации о состоянии системы за счет поглощения внутренних потоков Γ_i и замены аккумуляций в отдельных подсистемах одной суммарной величиной. Очевидно, такую агрегацию можно проводить повторно. Тем самым появляется возможность описания и взаимосвязанной оценки состояния ХТС на нескольких уровнях детализации, соответствующих, например, отдельному аппарату, технологическому отделению, цеху и т.д. Идея многоуровневой оценки состояния ХТС хорошо согласуется с иерархической структурой интегрированных АСУ.

Решение задачи оценки состояния начинается с наиболее детального уровня описания системы. Полученные оценки внешних потоков агрегированной подсистемы и суммарная аккумуляция (если она оценима) используются на следующем уровне как результаты измерения. Аналогичные переходы оценок состояния модели нижнего уровня в результаты измерения для следующего уровня агрегации системы происходят на всех уровнях описания ХТС.

При решении задачи оценки состояния ХТС в составе АСУ ТП низшим уровнем является уровень датчиков, где происходит фильтрация результатов измерения технологических параметров. Для этого используется ФК, базирующийся на одномерной модели "состояние – выход":

$$y(k+1) = y(k) + v(k)$$
$$z(k) = y(k) + w(k) \qquad (14)$$

где $y(k)$ и $z(k)$ - скалярные состояние и выход. Полученные из одномерного ФК оценки

$$\hat{y}(k+1) = \hat{y}(k) + K(k+1)\,[z(k+1) - \hat{y}(k)] \qquad (15)$$

используют в ФК, основанных на дискретных динамических или квазистационарных моделях (I), (2) как результаты измерений. При этом выборочные значения показаний датчиков $z(k+1)$ и обновляющих процессов $\delta(k+1) = z(k+1) - \hat{y}(k)$ применяют для диагностики датчиков и преобразователей, а также обнаружения нарушений технологического режима.

Литература

I. Аарна О.А. Динамическая балансовая модель химико-технологической системы и модели подсистем. - Тр. Таллинск. политехн. ин-та, 1982, № 538, с. 113-123.

2. Aarna O. Dynamic chemical plant model and its application. - In: Preprints of the Seventh Triennial World Congress of IFAC, Helsinki, Finland, 1978, v. 1, p. 279-286.

3. Anderson B.D.O., Moore J.B. Optimal Filtering. Englewood Cliffs, Prentice-Hall Inc., 1979. 357 p.

4. Аарна О.А. Обобщенная динамическая модель химико-технологической системы. - Тр. Таллинск. политехн. ин-та, 1981, № 519, с. 93-100.

5. Аарна О.А. Качественный анализ обобщенной динамической модели химико-технологической системы. - Тр. Таллинск. политехн. ин-та, 1981, № 519, с. 101-114.

O. Aarna

DECOMPOSITION OF THE CHEMICAL PLANT STATE ESTIMATION PROBLEM

Summary

Decomposition of the chemical plant state-space material and energy balance model and the plant state estimation problem is considered. From Kalman filter gain matrix block-diagonality two ways of decomposition: functional and structural decomposition are deduced and properties of the decomposed models are discussed. Multilevel state estimation schemes are described.

№ 550

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED

ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

УДК 66.011:681.3.06

Ю.Ю. Ребане

ГЕНЕРАТОР МОДУЛЕЙ АППАРАТОВ ХИМИКО-ТЕХНОЛОГИЧЕСКИХ СХЕМ

I. Введение

Из общих затрат времени на моделирование химико-технологических схем (ХТС) большую часть занимает разработка и программирование алгоритмов расчета отдельных аппаратов. Для ускорения программирования следует автоматизировать этот процесс. При автоматизации человек разрабатывает алгоритм расчета и вводит его в вычислительную машину, где специальный генератор составляет программные модули. Такой подход применен в системе моделирования ХТС PEETER-80.

Основные принципы построения системы PEETER-80 приведены в работах [1, 2, 3]. Система имеет диалог с пользователем, обеспечивающий ввод данных, изменение структуры моделируемой схемы и хода вычислений. Система позволяет решать широкий круг задач моделирования статики и динамики ХТС. Для моделирования надо составить модели аппаратов и схем в виде стандартных для системы PEETER-80 подпрограмм на ФОРТРАНе. Чтобы исключить труд квалифицированного программиста для программирования этих модулей к системе был разработан генератор программ PROGRAMMIST.

Генератор PROGRAMMIST предназначен для создания в режиме диалога программных модулей отдельных аппаратов или схем на языке ФОРТРАН, для их трансляции и последующего построения образа задачи моделирования. Генератор PROGRAMMIST состоит из двух частей:

1. Генератор модулей аппаратов (ГМ).
2. Генератор образа задачи моделирования (ГЗ).

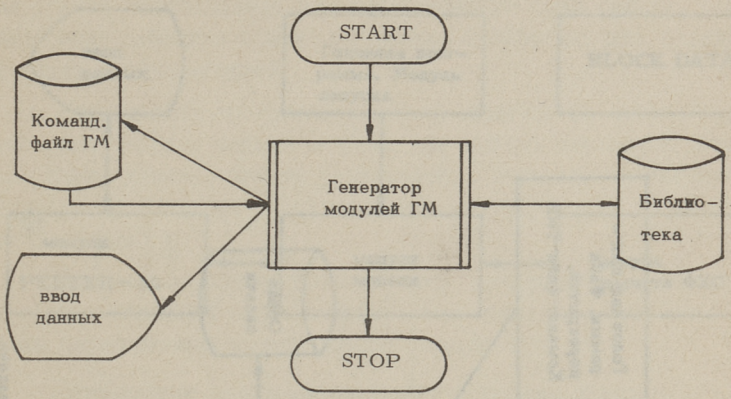
2. Генератор модулей (ГМ)

Структурная схема генератора модулей приведена на фиг. I. ГМ в диалоге запрашивает необходимые данные и генерирует модуль аппарата на языке ФОРТРАН, а также одноименный с модулем аппарата командный файл ГМ. Модуль транслируют и помещают в библиотеку моделей. Сгенерированный командный файл сохраняет все введенные данные. Для исправления выявленных ошибок запускают этот командный файл и генерируют исправленный модуль аппарата.

Для генерации вводятся имя и название сгенерированной подпрограммы, матрица схемы и текст алгоритма расчета аппарата. Дополнительно вводятся списки потоков, целых и вещественных переменных, индексов для указания итерируемых переменных. Задачей ГМ является оформление комментариев, интерфейса и операторов подпрограммы. По исходной информации генерируются операторы описания массивов, общих блоков, операторы эквивалентности и операторы DATA для задания численных и текстовых констант, обращения к системным программам.

Введенный текст алгоритма расчета, не анализируя, помещают в модуль в неизменном виде. В тексте допускаются выражения, алгебраические и дифференциальные уравнения, обращения к подпрограммам. Для облегчения создания модели к генератору модулей добавлен генератор уравнений (ГУ). При вводе текста можно вызывать ГУ и генерировать в исходный текст балансные уравнения (теплоты, масс ...), уравнения расчета равновесия фаз, обращения к модулям расчета физико-химических свойств. При этом балансные уравнения можно генерировать в явном виде, то есть когда они решены относительно некоторых переменных.

Пользователь описывает модель в своих терминах по правилам языка ФОРТРАН. Имеются некоторые ограничения. Не допускается использование некоторых идентификаторов. В системе PEETER-80 принят стандартный вид потока. При написании текста модели элементы потоков должны быть написаны в следующем виде: $P(L_{j+i})$. Здесь j - номер потока и i - номер элемента потока.

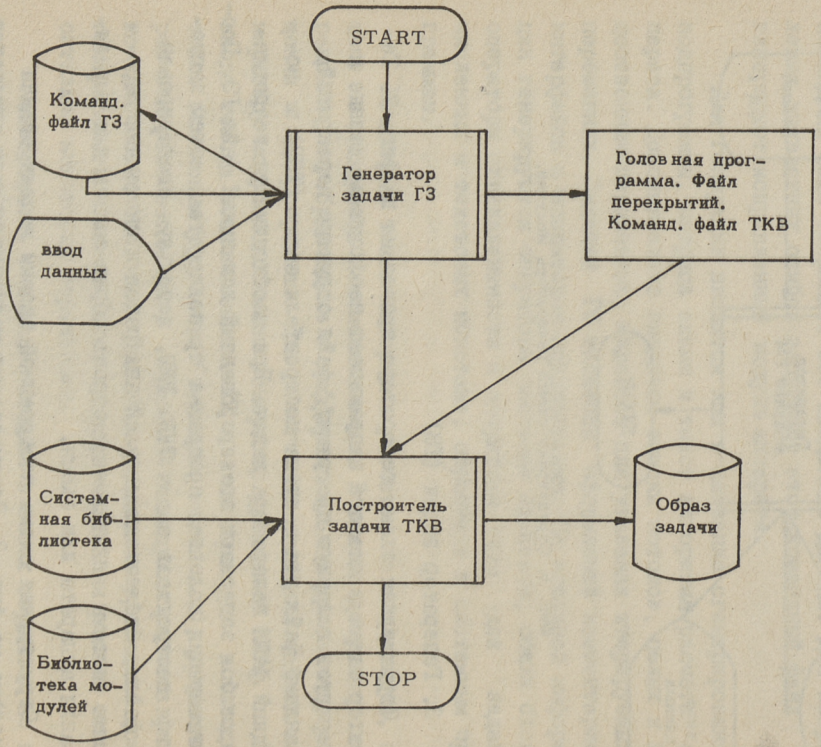


Фиг. 1. Структурная схема генератора модулей.

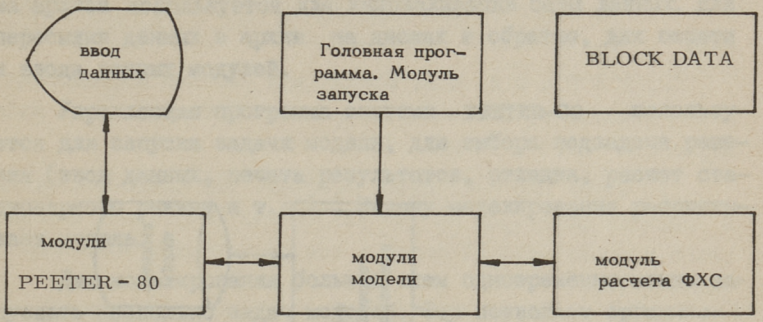
3. Генератор задач (ГЗ)

Структурная схема генератора приведена на фиг. 2. Генератор задач в диалоге запрашивает необходимые данные и генерирует головную программу, файл структур перекрытий, командный файл для запуска построителя задач (ТКВ) и командный файл генератора задач. При необходимости исправления ошибок запускают сгенерированный командный файл ГЗ. После генерации головная программа транслируется, затем запускается построитель задач ТКВ. ТКВ, используя сгенерированные файлы, библиотеки модулей аппаратов и системные программы системы PEETER-80, строит образ задачи для моделирования аппарата или схемы.

Структурная схема построенной задачи моделирования приведена на фиг. 3. Ядром задачи являются модули аппаратов и схем, которые обращаются к системным программам для ввода данных из базы данных или в режиме диалога. Для расчета физико-химических свойств веществ имеется отдельный модуль (ФХС). Начальные данные в общие области задают в BLOCK DATA. В головной программе задают имя схемы и имя модуля расчета ФХС. Модуль запуска организует запуск модели.



Фиг. 2. Структурная схема генератора задачи.



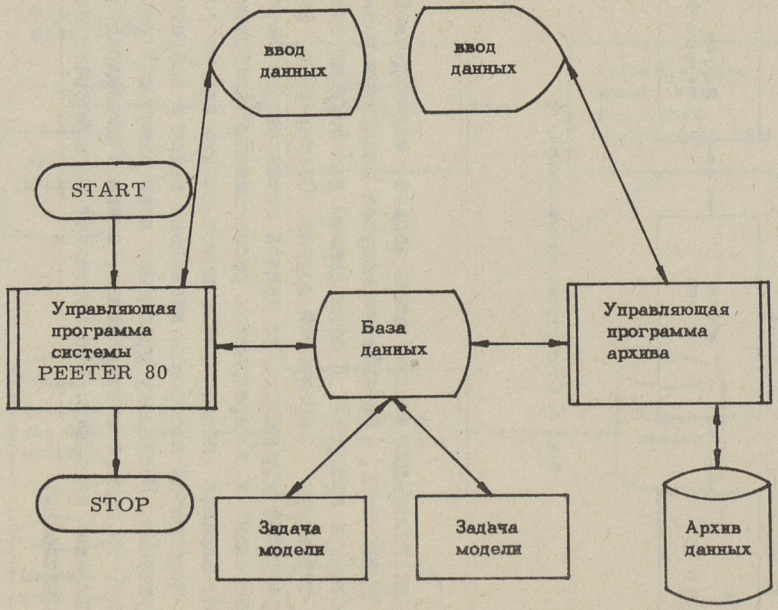
Фиг. 3. Структурная схема задачи.

Для генерации вводят имена задачи, моделируемой схемы, BLOCK DATA, библиотеки модулей аппаратов и библиотеки системных программ. К этим данным для больших схем вводят структуру перекрытий модели. Составление структуры перекрытий упрощено. Для каждой ветви вводят имена последовательно и с перекрытием расположенных программ и ветвей. По желанию пользователь может вместо задачи со структурой перекрытий построить несколько задач и запустить их одновременно. Поскольку уравнения всех аппаратов решают совместно, то при решении неважно, в какой последовательности рассчитывают аппараты. Это позволяет организовать параллельные вычисления.

4. Запуск задачи

В конце работы генератор задач запускает систему. При повторном решении запуск осуществляют вне генератора.

Схема работы системы PEETER-80 со сгенерированной задачей приведена на фиг. 4. Система использует базу дан-



Фиг. 4. Схема решения.

ных в оперативной памяти. База данных составлена в отдельном разделе памяти в общей области. Управляющая программа архива используется для инициализации базы данных, для пересылки данных в архив на дисках и обратно, для печати и ввода данных модулей.

Управляющая программа системы PEETER-80 используется для запуска задачи модели, для выбора подзадачи решения (ввод данных, печать результатов, отладка, расчет стационарного режима и т.д.) . Задачу моделирования рассчитывает модель.

Для моделирования больших схем одновременно можно запускать несколько задач моделей. Это позволяет организовать параллельные вычисления на разных ЭВМ.

Управляющие программы работают в режиме диалога.

5. Выводы

Применение генератора PROGRAMMIST позволяет провести весь процесс моделирования ХТС в режиме диалога. Опыт эксплуатации генератора PROGRAMMIST показал, что время на программирование и на отладку моделей аппаратов и схем сокращается в несколько раз. Разработчик освобождается от проблем программирования. Составление модулей происходит на уровне алгоритма. Оформление текста расчета алгоритма требует незначительных знаний по программированию. Кроме этого, генерация балансных уравнений сокращает объем разрабатываемого алгоритма. Система PEETER-80 освобождает пользователя также от составления операторов ввода данных, печати результатов и от организации хода решения. Система позволяет свободно оперировать моделью: решать целую схему совместно или выбирать отдельные аппараты схемы, изменять подзадачи решения и проводить отладку.

Система PEETER-80 работает на ЭВМ СМ-4 и ЕС. Генератор PROGRAMMIST работает на ЭВМ СМ-4 и использует процессор косвенных командных файлов. Сгенерированные модули аппаратов и схем на языке ФОРТРАН можно использовать и на других ЭВМ.

Перенос осуществляется на магнитной ленте или на перфоленте.

Литература

I. Ребане Ю.Ю. Моделирование СХТС в режиме диалога - Тр. Таллинск. политехн. ин-та, 1982, № 538,

2. Ребане Ю.Ю. Прикладная система моделирования ХТС. - Тезисы всесоюзной конференции "Математическое моделирование СХТС". Таллин, 1982.

3. Отчет НИР ТПИ. Пакет программ для расчета статики и динамики реакторов КС для разложения фосфогипса и ковдорского концентрата. УІ часть.№ гос.регистрации 76092835. Таллин, 1981.

J. Rebane

PROGRAM MODULE GENERATOR FOR CHEMICAL PLANT MODELLING

Summary

The program generator PROGRAMMIST of the modelling system PEETER-80 is described. The PROGRAMMIST generates FORTRAN subroutines for unit operation and chemical plant modelling. The model and task-building proceeds in dialogue with the user. In principle, parallel processing of the modelling problem on several computers is possible. The module generator has been implemented on a CM-4 minicomputer. The modelling system PEETER-80 is available on ES and CM-4 computers.

№ 550
TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED
ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА
УДК 621.316.726.078

М.В. Мин

ДИНАМИЧЕСКИЙ СИНТЕЗ ВЫСОКОТОЧНЫХ СЛЕДЯЩИХ СИСТЕМ

I. Введение

Динамическим называем синтез, проводимый с целью достижения наперед заданных динамических показателей системы [I], важнейшими среди которых являются погрешность слежения и длительность переходных процессов. Для минимизации погрешности слежения за изменяющимися во времени входными воздействиями, применяют системы с астатизмом второго порядка со следующей типичной передаточной функцией $W(s)$ третьего порядка разомкнутой системы [I-3]:

$$W(s) = \frac{T_2 s + 1}{\tau_a^2 s^2 (T_3 s + 1)}, \qquad (I)$$

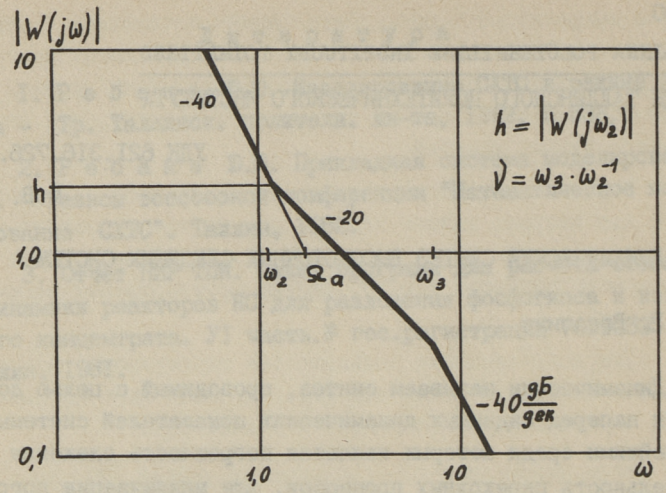
логарифмическая амплитудно-частотная характеристика (ЛАЧХ) которой приведена на фиг. I.

2. Постановка задачи

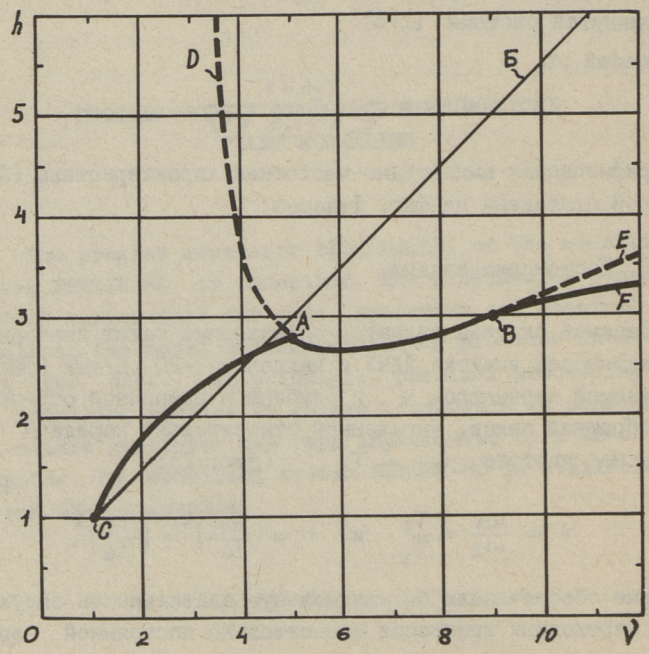
Задачей синтеза является определение таких значений для протяженности участка ЛАЧХ с наклоном -20 дБ/дек (фиг. I), выражаемой параметром ν, и глубины h единичной отрицательной обратной связи, выраженной относительно передачи системы на частоте $\omega_2 = T_2^{-1}$ (фиг. I):

$$\nu = \frac{\omega_3}{\omega_2} = \frac{T_2}{T_3} \quad \text{и} \quad h = \left(\frac{\Omega_a}{\omega}\right)^2 = \left(\frac{T_2}{\tau_a}\right)^2, \qquad (2)$$

которые обеспечивали бы минимальную длительность затухания $t_з$ переходных процессов относительно постоянной времени τ_a:



Фиг. 1. Логарифмическая амплитудно-частотная характеристика (ЛАЧХ) разомкнутой следящей системы.



Фиг. 2. Зависимости $h = f(\nu)$ при разных частных случаях.

$$\Theta_з = \frac{t_з}{\tau_a} = \Omega_a t_з. \qquad (3)$$

Для определения $\Theta_з$ воспользуемся следующими выражениями в нормированном времени $\Theta = \Omega_a t$:

$$x_1(\Theta) = A_1 e^{-\alpha_1 \Theta} + A_2 e^{-\alpha_2 \Theta}, \qquad (4)$$

$$x_2(\Theta) = G_1 e^{-\gamma_1 \Theta} + G_2 e^{-\gamma_2 \Theta} + G_3 e^{-\gamma_3 \Theta}, \qquad (5)$$

где α_1 и α_2 – нормированные вещественный полюс и вещественная часть комплексно-сопряженных полюсов;

$\gamma_1; \gamma_2; \gamma_3$ – соответственные вещественные полюса.

Выражения (4) и (5) описывают затухания огибающих соответственно колебательных и апериодических переходных процессов относительно их установившихся значений, причем при аргументе $\Theta = \Theta_з$ величины $x_1(\Theta)$ и $x_2(\Theta)$ получают значение допустимой ошибки.

Известно [3], что при $\varepsilon \leq 0{,}05$ длительность переходных процессов слабо зависит от значений коэффициентов A и G экспоненциальных функций в суммах (4) и (5), а у высокоточных систем ($\varepsilon \leq 0{,}01$) эта зависимость уже ничтожна. Поэтому можно принять во внимание только значения α_1 и α_2 или γ_1; γ_2; γ_3. Отметим еще, что скорость затухания сумм (4) и (5) определяет в конечном счете тот член суммы, который затухает медленнее остальных.

На основе вышеизложенного можно сделать вывод, что синтез сводится к определению оптимальных значений для параметров h и ν, которые максимизировали бы наименьшее значение среди α_1 и α_2 или $\gamma_1; \gamma_2; \gamma_3$, которое играет решающую роль при определении темпа затухания функций (4) и (5).

3. Основы метода синтеза

Для нахождения основ синтеза необходимо найти аналитические решения характеристического уравнения H(s) замкнутой системы, составленной на основе выражений (I) и (2):

$$H(s) = \frac{\sqrt{h}}{\nu} \tau_a^3 s^3 + \tau_a^2 s^2 + \sqrt{h}\, \tau_a s + 1. \qquad (6)$$

В общем случае уравнение (6) не имеет достаточно наглядных решений, связывающих корни уравнения с параметрами h и ν, но оказывается, что в данном частном случае – малейшее

значение среди вещественных корней и вещественных частей комплексных корней должно иметь максимально достижимое значение, задача упрощается решительно. На фиг. 2 приведены графики, изображающие функции $h = f(\nu)$ при выполнении вышеизложенного условия.

Линия ДВ ($\nu = 3...9$) описывается выражением:

$$h = \frac{2\nu^2}{9\nu - 27} \tag{7}$$

и определяет такие значения h, которые обеспечивают равенство вещественного корня α_1 и вещественной части α_2 пары комплексных полюсов ($\alpha_1 = \alpha_2$), причем в пределах отрезков АВ ($\nu = 5...9$) выполняется при этом также условие оптимальности:

$$h_{опт} = \frac{2\nu^2}{9\nu - 27}\Bigg|_{\nu = 5...9}, \tag{8}$$

которое гарантирует максимальное значение α_{AB} для $\alpha_1 = \alpha_2$:

$$\alpha_{AB} = -\sqrt{\frac{\nu - 3}{2}}. \tag{9}$$

Остальные параметры, получаемые на основе корней уравнения (6), следующие:

$$\omega_к = \sqrt{\frac{9 - \nu}{2}}; \quad \omega_n = \sqrt{3}; \quad \xi = \sqrt{\frac{\nu - 3}{6}}, \tag{10}$$

где $\omega_к$ – частота колебаний переходных процессов (мнимая часть пары комплексных полюсов);

ω_n – собственная частота колебаний (модуль пары комплексных полюсов);

ξ – коэффициент затухания колебаний ($\xi = \alpha_1 \omega_n^{-1}$).

При $\nu < 5$ (линия АС на фиг. 2) вещественная часть α_2 комплексных полюсов меньше значения вещественного полюса ($\alpha_2 < \alpha_1$) при $h = h_{опт}$ и поэтому выражение:

$$h_{опт} = \left(\frac{2\nu}{\nu + 1}\right)^2\Bigg|_{\nu = 1...5} \tag{11}$$

обеспечивает для α_2 максимально возможное значение

$$\alpha_{AC} = -\frac{\nu - 1}{4}, \tag{12}$$

причем вещественный корень α_1 получает постоянное значение:

$$\alpha_1 = -1, \tag{13}$$

а остальные параметры выражаются в следующем виде:

$$\omega_к = \sqrt{\frac{-\nu^2+10\nu+7}{16}};\quad \omega_n = \sqrt{\frac{\nu+1}{2}};\quad \xi = \frac{\nu-1}{4}\sqrt{\frac{2}{\nu+1}}. \qquad (I4)$$

При $\nu \geq 9$ (линии BE и BF на фиг. 2) в оптимальном случае два из трех вещественных полюсов γ_1; γ_2; γ_3 совпадают и соответствующие значения h определяются выражением:

$$h = \frac{\nu(\nu+18)-27}{8\nu} \pm \sqrt{\left(\frac{\nu(\nu+18)-27}{8\nu}\right)^2 - \nu}, \qquad (I5)$$

причем знак "-" перед квадратным корнем соответствует выражению для оптимального значения $h = h_{опт}$ (линия BF) и знак "+" соответствует выражению, описывающему линию BE на фиг.2. Отметим, что совпадающие между собой малейшие два ($\gamma_2 = \gamma_3$) из трех вещественных полюсов имеют при $h = h_{опт}$ максимально возможное значение, которое постепенно убывает при увеличении ν и получает предельное единичное значение при $\nu = \infty$ и $h_{опт} = 4$.

Выражения (9) и (I2) позволяют определить, что малейшие вещественные полюс или вещественная часть комплексных полюсов имеют при $h = h_{опт}$ следующее экстремальное значение:

$$\text{макс}\,\alpha = \sqrt{3} \quad \text{при} \quad \begin{cases} \nu = 9 \\ h = 3 \end{cases}, \qquad (I6)$$

которому на фиг. 2 соответствует точка B, характеризуемая совпадением всех трех вещественных полюсов ($\gamma_1 = \gamma_2 = \gamma_3$). Отметим, что выражение (I6) позволяет определить точные значения параметров для оптимальной по быстродействию следящей системы с передаточной функцией (I) разомкнутого состояния, при условии, что допустимая ошибка приближается к бесконечно малому значению ($\varepsilon \to 0$).

4. Выводы.

Прямая Б на фиг. 2 изображает функцию $h_Б = f(\nu)$, полученную в результате применения метода динамического синтеза В.А. Бесекерского [I]. Необходимо отметить, что эти результаты существенно отличаются от полученных выше значений параметров, в частности, при $\nu > 5$. Кроме этого, полученные нами результаты не утверждают также рекомендацию выбрать $\nu > I0$ [3]. Это дает нам основу сомневаться в целесообраз-

ности применения этих хорошо известных методов [I, 3] при синтезе высокоточных следящих систем.

Литература

I. Бесекерский В.А. Динамический синтез систем автоматического регулирования. М., Наука, 1970. 576 с.

2. Кийве Р.Х., Мин М.В., Паавле Т.Э. Динамический синтез системы ФАП третьего порядка. - Тезисы докладов Республиканской научно-технической конференции "Современные методы и устройства радиоэлектронного оборудования", посвященной Дню радио. Секция "Информационно-измерительная техника". Таллин, 1981, с. 9-10.

3. Техническая кибернетика. Теория автоматического регулирования. Кн. I. Математическое описание, анализ устойчивости и качества систем автоматического регулирования / Под ред. В.В. Солодовникова. М., Машиностроение, 1967. 770 с.

M. V. Min

DYNAMIC SYNTHESIS OF THE PRECISION TRACKING SYSTEMS

Summary

In this paper there is presented the method of the dynamic synthesis for the third-order high precision astatic tracking systems. The analytical formulae enabling to identify the values of the parameters of the system giving the fastest damping of the transient processes are given. It is stated the given method is much more precise than the other widely known methods, especially, if the predetermined values of the dynamic errors are very small.

№ 550
TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED
ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА
УДК 681.511.26

К.А. Райенд

ДЕКОМПОЗИЦИЯ ЛИНЕЙНОЙ МОДЕЛИ ОБЪЕКТА УПРАВЛЕНИЯ

I. Введение

Обозначим через (A, B, C) модель объекта управления, которая задается уравнениями

$$\dot{x} = Ax + Bu, \quad x_0$$
$$y = Cx,$$

где x - вектор состояний размерности n,
u - вектор входов размерности m,
y - вектор выходов размерности r,
A,B,C - постоянные матрицы размерности $n \times n, n \times m, r \times n$ соответственно.

Многие физические объекты управления имеют несколько входов и выходов. В действительности, влияние одного входа на какой-то выход может быть гораздо слабее, чем влияние другого входа на этот же выход. Методы построения модели не способны выявить слабо связанные входы и выходы и поэтому получаются модели чрезмерно высокого порядка.

В действительности, многие связи вход-выход несущественные и без большого ущерба можно декомпонировать одну связанную модель на несколько автономных моделей, которые описывают соответствующие сильно связанные входы и выходы. Определение сильно связанных вход-выход пар приведено в[I].

В общем случае возникают следующие вопросы, решению которых посвящена статья:

- как обнаружить наличие слабо связанных входов и выходов;
- на сколько подмоделей распадается исходная модель объекта управления;
- какие входы - состояния - выходы принадлежат соответствующим подмоделям;

- как представить уравнения подмоделей;
- какую ошибку влечет за собой декомпозиция исходной модели на автономные подмодели.

2. Основные понятия

При анализе управляемости и наблюдаемости объекта используют матрицы Грама [2]. Грамиан управляемости имеет вид

$$W_c^2 = \int_0^{t_1} e^{At} B B^T e^{A^T t} dt,$$

а грамиан наблюдаемости

$$W_o^2 = \int_0^{t_1} e^{A^T t} C^T C e^{At} dt.$$

Обозначим сингулярное разложение [3] матриц W_c^2 и W_o^2,

$$W_c^2 = U_c \Sigma_c^2 U_c^T, \qquad W_o^2 = U_o \Sigma_o^2 U_o^T, \tag{I}$$

где U_c и U_o ортонормированные матрицы и $\Sigma_c = diag(\sigma_{cj})$, $\Sigma_o = diag(\sigma_{oj})$, $1 \le j \le n$. Сингулярные значения σ_{cj} называются величинами главных компонентов (ВГК) матрицы $e^{At}B$ на промежутке $[0, t_1]$ [4].

Матрицы Грама зависят от преобразования координат состояния. Существует такое преобразование состояния, что

$$W_c^2 = W_o^2 = \Lambda^2, \tag{2}$$

где $\Lambda = diag(\lambda_j)$, $1 \le j \le n$.

Модель называется балансированной моделью, если имеет место равенство (2).

Импульсные переходные функции равняются $e^{At}B$. Диагональные элементы грамиана управляемости можно интерпретировать как нормы импульсных переходных функций

$$w_{jj}^2(W_c^2) = \int_0^{t_1} (x_j)^2 dt,$$

где x_j обозначают j-ый элемент вектора состояния и $w_{jj}(W_c^2)$ обозначает диагональный элемент грамиана W_c^2.

Объект (A, B, C) является полностью управляемым и полностью наблюдаемым, если матрицы W_c и W_o имеют полный ранг [2]. Из равенств (I) видно, что управляемость и наблюдаемость ппределены ВГК σ_{cj} и σ_{oj} соответственно.

Если среди ВКГ σ_{cj} или σ_{oj} существует такой, который равняется нулю, то объект (A, B, C) неуправляемый и ненаблюдаемый. В практических расчетах часто даже для теоретически неуправляемого объекта минимальный среди σ_{cj} не равняется точно нулю. Причиной являются ошибки округления в расчетах.

В действительности представляет интерес определение эффективной управляемости и наблюдаемости. Обозначаем минимальное и максимальное среди ВГК σ_{cj} через σ_{cmin} и σ_{cmax}. Мерой управляемости называем отношение $\mu_c = \sigma_{cmin} / \sigma_{cmax}$. Объект называем эффективно управляемым, если $\mu_c > \varepsilon$, где ε заданное число, $\varepsilon < 1$. Аналогичным образом определяется мера наблюдаемости. Для балансированной модели меры управляемости и наблюдаемости равны.

Мера управляемости балансированной модели характеризует относительное влияние входных воздействий на отдельные состояния. Например, если $\mu_c = 10^{-4}$, то это означает, что существует состояние балансированной модели, на которое влияют входы в 10^4 раза слабее, чем максимальное влияние. Для балансированной модели можно, благодаря равенству (2), сказать, какие состояния хорошо, какие плохо управляемы.

Пусть U обозначает множество всех входов. Делим множество U на p_u непересекающихся подмножеств входов и и называем такое подмножество группой входов. Каждую i-ую группу входов рассматриваем как один вход и обозначаем через U_i, $1 \le i \le p_u$. Дополнение группы входов U_i относительно множества входов обозначаем через $\bar{U}_i$, $U_i \cup \bar{U}_i = U$. Подмножество состояний, управляемых входами из группы входов U_i, обозначаем через $\mathcal{X}_{u_i}$. Аналогично делим множество выходов y на p_y непересекающихся подмножеств выходов и называем такое подмножество группой выходов y_k, $1 \le k \le p_y$. Пусть $\mathcal{X}_{y_k}$ обозначает подмножество состояний, наблюдаемых из группы выходов y_k.

3. Основная идея

Подаем на входы модели (A, B, C) импульсные воздействия $u^i(t) = e_i \delta(t)$, $1 \le i \le m$, где e_i равняется i-му столбцу $m \times m$ единичной матрицы и $\delta(t)$ скалярная импуль-

сная функция. Соответствующая импульсная переходная функция равняется

$$x^i(t) = e^{At} b_i, \quad x_0 = 0,$$

где b_i i-ый столбец матрицы B, $B = (b_1, b_2, \ldots, b_m)$.

Рассматриваем матрицы Грама по входам

$$W^2_{c,i} = \int_0^{t_1} e^{At} b_i b_i^T e^{A^T t} dt. \qquad (3)$$

Если модель объекта дана в балансированной форме, то

$$\sum_{i=1}^{m} W^2_{c,i} = \Lambda^2.$$

Следовательно,

$$\int_0^{t_1} (x_j)^2 dt = \sum_{i=1}^{m} \int_0^{t_1} (x_j^i)^2 dt,$$

$$\int_0^{t_2} (x_j)^2 dt = \sum_{i=1}^{m} w_{jj}^2 (W^2_{c,i}),$$

где x_j j-ая компонента вектора состояния и $w_{jj}^2 (W^2_{c,i})$ обозначает диагональный элемент грамиана $W^2_{c,i}$.

Идея определения влияния отдельных входов на состояние состоит в следующем. Рассмотрим отношения:

$$\frac{w_{jj}(W^2_{c,i})}{\lambda_j}, \quad i \leq j \leq n, \quad 1 \leq i \leq m. \qquad (4)$$

Если значение отношения (4) относительно малое, то i-й вход добавляет в импульсную переходую функцию j-го компонента состояния балансированной модели незначительную долю.

Данный подход имеет недостаток, вызванный несравнимостью единиц измерения входных воздействий в модели (A,B, C). Если мы во вход подаем импульс $e_i \delta(t)$, то он имеет единицу измерения, которую имели в виду для данного входа при составлении модели. Изменение единицы измерения соответствует умножению столбца b_i на скаляр α_i. Грамиан управляемости для i-го входа тогда равняется $\tilde{W}^2_{c,i} = \alpha_i^2 W^2_{c,i}$ и $w_{jj}(\tilde{W}^2_{c,i}) = \alpha_i \cdot w_{jj}(W^2_{c,i})$. Соответственно меняется значение отношения (4).

Следовательно, следует нормировать входы (столбцы b_i). Аналогичным образом исследуется влияние отдельных состояний на выходы. При этом удобно использовать свойство дуальности линейной модели [2].

4. Нормирование входов и выходов

Подаем на i-й вход модели импульсную функцию. Вычисляем соответствующие матрицы грама $W^2_{c,i}$, $1 \le i \le m$. ВГК вектора $e^{At}b_i$ обозначаем через σ^i_{cj}, $1 \le j \le n$ и максимальный среди них через $\sigma^i_{c\,max}$.

Естественно нормировать входы модели так, чтобы влияние отдельных входов на состояния было равно. Берем за основу нормирования максимальные ВГК $\sigma^i_{c\,max}$.

О п р е д е л е н и е. Модель (A, B, C) назовем моделью с нормированными входами на промежутке $[0, t_1]$, если

$$\sigma^1_{c\,max} = \sigma^2_{c\,max} = \ldots = \sigma^m_{c\,max}.$$

Геометрически это означает, что наибольшие полуоси эллипсоидов достигаемых точек в пространстве состояний моделей с одним входом равны между собой.

Для нормирования входов вычислим грамианы управляемости при каждом входе $W^2_{c,i}$, $1 \le i \le m$ и наибольшие ВГК $\sigma^i_{c\,max}$. Найдем наибольшую из них

$$\sigma_{c\,max} = \max_{1 \le i \le m} \sigma^i_{c\,max}.$$

Затем вычислим коэффициенты нормирования

$$\alpha_i = \frac{\sigma_{c\,max}}{\sigma^i_{c\,max}}, \quad 1 \le i \le m.$$

Столбцы нормированной матрицы входа $\tilde{B}$ равняются $\tilde{b}_i = \alpha_i b_i$.

Аналогично нормируются выходы модели (строки матрицы C), при использовании максимальных ВГК $\sigma^k_{o\,max}$ матриц $e^{A^T t} c^T_k$, $1 \le k \le r$.

Балансированная модель после нормирования входов и выходов не остается балансированной. В общем случае α_i не равны между собой и $\sum_{i=1}^{m} \alpha_i^2 W^2_{c,i}$ не будет диагональной матрицей.

О п р е д е л е н и е . Модель (A, B, C) назовем балансированной нормированной (БН) моделью, если

$$W_c = W_o = \Lambda$$

$$\sigma^1_{c\,max} = \sigma^2_{c\,max} = \ldots = \sigma^m_{c\,max}$$

$$\sigma^1_{o\,max} = \sigma^2_{o\,max} = \ldots = \sigma^r_{o\,max}$$

Укажем алгоритм нахождения БН модели. Нормирование входов состоит в преобразовании матрицы входа $\tilde{B} = B\mathcal{O}$, где $\mathcal{O} = diag(\alpha_1, \alpha_2, \ldots, \alpha_m)$. При нормировании выходов матрица выхода умножается слева на диагональную матрицу $\tilde{C} = \mathcal{B}C$, где $\mathcal{B} = diag(\beta_1, \beta_2, \ldots, \beta_n)$.

Обозначим модель, у которой проведено преобразование состояния и нормирование входов и выходов через $(A(T), B(T, \mathcal{O}), C(T, \mathcal{B}))$,

$$A(T) = T^{-1}AT$$

$$B(T, \mathcal{O}) = T^{-1}B\mathcal{O}$$

$$C(T, \mathcal{B}) = \mathcal{B}CT.$$

Приведем модель с преобразованием состояния T_1 в балансированную форму $(A(T_1), B(T_1, I_m), C(T_1, I_n))$. Нормируем входы и выходы, получая модель $(A(T_1), B(T_1, \mathcal{O}_1), C(T_1, \mathcal{B}_1))$. Как мы отметили, полученная таким образом модель небалансированная. Балансируем заново, получая балансированную модель $(A(T_2), B(T_2, \mathcal{O}_1), C(T_2, \mathcal{B}_1))$.

Вновь вычислим грамианы $W_{c,i}^2$ и $W_{o,i}^2$ и максимальные ВГК $\sigma_{c\,max}^i$ и $\sigma_{c\,max}^k$, $1 \le i \le m$, $1 \le k \le r$. Если окажется, что $|\sigma_{c\,max}^i - \sigma_{c\,max}^l| > \varepsilon$, $1 \le i$, $l \le m$ или $|\sigma_{o\,max}^k - \sigma_{o\,max}^l| > \varepsilon$, $1 \le k$, $l \le r$, то модель в новой внутренней системе координат вновь следует нормировать. Получаем модель $(A(T_2), B(T_2, \mathcal{O}_2), C(T_2, \mathcal{B}_2))$.

Следующая теорема [5] утверждает, что такая процедура балансирования и нормирования входов и выходов сходится.

Т е о р е м а I. Последовательное балансирование и нормирование модели сходится в БН модель

$$\lim_{k \to \infty} (A(T_k), B(T_k, \mathcal{O}_k), C(T_k, \mathcal{B}_k)) = (A_b, B_b, C_b),$$

где (A_b, B_b, C_b) - БН модель.

5. Внутренне автономные входы, выходы и состояния

Рассмотрим БН модель объекта (A_b, B_b, C_b) и выясним, как влияют на состояния отдельные входы. Целью выяснения является выделение таких состояний, которые независимы от некоторых входов. Так как модель нормированная, можем срав-

нивать влияние отдельных входов, а балансированность модели позволяет связывать влияние на состояния с диагональными элементами грамиана $W^2_{c,i}$.

О п р е д е л е н и е. Если найдется такая группа входов u_i, что множество управляемых ею состояний x_{u_i} не пересекается со множеством состояний $x_{\bar{u}_i}$, управляемых дополнением группы входов u_i, $x_{u_i} \cap x_{\bar{u}_i} = \phi$, то группу входов назовем внутреннеавтономной группой (ВАГ) входов. Соответствующее множество состояний x_{u_i} назовем ВАГ состояний со стороны входа.

Множество всех входов u всегда образует ВАГ входов.

Существование более одной ВАГ входов показывает, что со стороны входов можно модель объекта разделить на подмодели. ВАГ входов является множеством входов подмодели, а состояниями подмодели будут связанные с этой группой входов состояния. Так выглядит декомпозиция модели со стороны входа. Со стороны выхода декомпозиция может выглядеть совсем иначе.

Как выяснить существование ВАГ входов и состояний? Предварительно нужен критерий, позволяющий судить о значимости влияния входа на состояние.

К р и т е р и й I. Состояние x_j БН модели является сильно связанным с входом u_i, если

$$\frac{w_{jj}(W^2_{c,i})}{\lambda_j} > \varepsilon, \qquad (5)$$

где $w_{jj}(W^2_{c,i})$ обозначает квадратный корень диагонального элемента грамиана $W^2_{c,i}$,

λ_j - ВГК БН модели и

ε - заданное малое число.

Отношение (5) меняется в пределах от нуля до единицы. Выбором числа ε определяем, какое влияние считать существенным. Например, при $\varepsilon = 10^{-3}$ влияние в 10^3 раза меньше, чем максимально возможное, считаем несущественным.

Благодаря балансированности модели, диагональные элементы $w_{jj}(W^2_{c,i})$ отражают вклад каждого входа в j-ю ВГК λ_j, которая равняется нормой j-го состояния на про-

межутке $[0,t_1]$. Благодаря нормированности входов $w_{jj}(W^2_{c,i})$ сравнимы с влиянием i-го входа на j-е состояние.

Алгоритм определения ВАГ входов.

I. Вычисляется БН модель объекта.

2. Вычисляют грамианы $W^2_{c,i}$, $1 \le i \le m$.

3. Образуется $n \times m$ матрица инциденций $R=(r_{st})$, которая указывает на связанность отдельных входов состояниями. Элемент r_{st} принимает значение единицы, если t-й вход сильно связан с s-м состоянием на основе критерия I. В противном случае $r_{st}=0$.

4. Образуем симметричную $m \times m$ матрицу $S=R^TR$. Диагональный элемент s_{ii} равняется числу состояний, которые связаны с входом u_i. Элемент $s_{ik}=0$, если входы u_i и u_k внутренне не связаны.

5. Перестановкой строк и столбцов матрицы S преобразуем ее в блочно-диагональную форму $\tilde{S}=\mathrm{diag}(\tilde{S}_i)$, $1 \le i \le p_u$. Число диагональных блоков матрицы $\tilde{S}$ равняется числу ВАГ входов p_u. Входы, соответствующие диагональному блоку $\tilde{S}_i$, образуют ВАГ входов. Размерность каждого блока $\tilde{S}_i$ равняется числу элементов в группе входов U_i.

ВАГ состояний со стороны входа $\mathcal{X}_{u_i}$ легко определить из матрицы инциденций R.

О п р е д е л е н и е. Если найдется такая группа выходов y_k, что множество наблюдаемых ее состояний $\mathcal{X}_{y_k}$ не пересекается со множеством состояний $\mathcal{X}_{\bar{y}_k}$, наблюдаемых дополнением группы выходов y_k, $\mathcal{X}_{y_k} \cap \mathcal{X}_{\bar{y}_k}=\phi$, то группу выходов назовем ВАГ выходов. Соответствующее множество состояний $\mathcal{X}_{y_k}$ назовем ВАГ состояний со стороны выхода.

К р и т е р и й 2. Состояние x_j БН модели является сильно связанным с выходом y_k, если

$$\frac{w_{jj}(W^2_{o,k})}{\lambda_j} > \varepsilon, \qquad (6)$$

где $w_{jj}(W^2_{o,k})$ – обозначает квадратный корень диагонального элемента грамиана $W^2_{o,k}$,

λ_j – ВГК модели и

ε – заданное малое число.

Алгоритм определения ВАГ выходов аналогичен алгоритму определения ВАГ входов. В вычислениях можно использовать программу определения ВАГ входов, заменяя в исходных данных матрицу A на матрицу A^T и матрицу B на C^T.

6. Декомпозиция модели

У модели могут существовать ВАГ входов и ВАГ выходов, которым соответствуют ВАГ состояний со стороны входа и со стороны выхода. Может оказаться, что множества состояний $\mathfrak{X}_{U_i}$ и $\mathfrak{X}_{y_k}$ равны. Тогда модель имеет автономную подмодель со входами U_i, состояниями $\mathfrak{X}_i$, $\mathfrak{X}_i = \mathfrak{X}_{U_i}$ и выходами y_k.

О п р е д е л е н и е. Если у модели существуют ВАГ входов U_i и ВАГ выходов y_k такие, что $\mathfrak{X}_{U_i} = \mathfrak{X}_{y_k}$, то соответствующие группы входов U_i и выходов y_k назовем внутренне сильно связанными.

Модель объекта, которая содержит внутренне сильно связанные группы входов и выходов, распадается на автономные подмодели. Каждая подмодель связывает одну группу входов с одной группой выходов через соответствующую группу состояний.

Алгоритм определения декомпонируемости модели состоит из следующих действий.

1. Определение ВАГ входов U_i и ВАГ состояний со стороны входа $\mathfrak{X}_{U_i}$.

2. Определение ВАГ выходов y_k и ВАГ состояний со стороны выхода $\mathfrak{X}_{y_k}$.

3. Согласование ВАГ входов и выходов.

7. Согласование групп входов и выходов.

При определении ВАГ входов мы не интересуемся выходами. Аналогично при определении ВАГ выходов не рассматриваются входы. Впоследствии ВАГ состояний со стороны входа и со стороны выхода могут оказаться неравными (несогласованными). Пример такой ситуации приведен на фиг. 1. Входы u_1 и u_2 образуют ВАГ входов

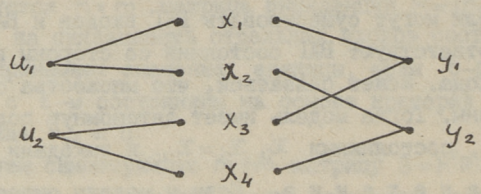
$$U_1 = \{u_1\}, \quad \mathfrak{X}_{U_1} = \{x_1, x_2\}$$

$$U_2 = \{u_2\},\ \mathfrak{X}_{U_2} = \{x_3\ x_4\}.$$

Выходы y_1 и y_2 образуют ВАГ выходов

$$Y_1 = \{y_1\},\ \mathfrak{X}_{Y_1} = \{x_1\ x_3\}$$

$$Y_2 = \{y_2\},\ \mathfrak{X}_{Y_2} = \{x_2\ x_4\}$$



Фиг. 1.

Хотя существуют две ВАГ входов и две ВАГ выходов, нет внутренне сильно связанных групп входов и выходов, так как соответствующие группы состояний не равны.

Согласование ВАГ входов и выходов происходит итеративно выполнением на каждой итерации двух шагов:

1) согласование ВАГ выходов со стороны входа;

2) согласование ВАГ входов со стороны выхода.

Два шага повторяются, пока группы входов и выходов согласованы.

Согласование ВАГ выходов со стороны входа состоит в следующем. Рассматривают поочередно все ВАГ входов. В каждой ВАГ входов U_k просматривают все входы u_i, $u_i \in U_k$. Для данного входа u_i рассматривают, на какие состояния x_j вход u_i имеет влияние, $x_j \in \mathfrak{X}_{U_k}$. Это же состояние x_j принадлежит в какой-то ВАГ состояний со стороны выхода $x_j \in \mathfrak{X}_{Y_\ell}$. Состояние x_j связано с выходами y_s, $y_s \in Y$. Если все выходы y_s принадлежат одной ВАГ выходов, скажем Y_ℓ, то ВАГ входов U_k и ВАГ выходов Y_ℓ согласованы со стороны входа. Если выходы y_s принадлежат различным ВАГ выходов Y_{ℓ_1} и Y_{ℓ_2}, то эти группы выходов надо объединить в одну группу Y_ℓ, $Y_\ell = Y_{\ell_1} \cup Y_{\ell_2}$, $\mathfrak{X}_{Y\ell} = \mathfrak{X}_{Y\ell_1} \cup \mathfrak{X}_{Y\ell_2}$.

Согласование ВАГ входов со стороны выхода происходит аналогично. Рассматривают все выходы по группам выходов и объединяются при необходимости группы входов.

Признаком согласованности является то, что при согласовании входов и выходов не надо было объединять ни одной группы.

Итеративный процесс согласования обязательно заканчивается. При каждой итерации уменьшается число ВАГ входов и/или выходов или число групп остается неизменным. Число ВАГ входов и выходов может уменьшаться до единицы. Но модель с одним ВАГ входов и одним ВАГ выходов является исходной моделью, и она согласована.

В процессе согласования число ВАГ входов и число ВАГ выходов приравниваются. Обозначаем число ВАГ входов и выходов после согласования через p.

8. Уравнения подмоделей

Если после согласования ВАГ входов и выходов оказывается, что число групп входов и выходов больше единицы, то модель объекта распадается на p автономных подмоделей. Рассмотрим, как получить уравнения подмоделей.

Имеем БН модель (A_b, B_b, C_b) и внутренне сильно связанные группы входов, состояний и выходов (u_i, x_i, y_i), $1 \le i \le p$. Переставим входы, состояния и выходы БН модели так, чтобы входы, состояния и выходы одной группы располагались подряд.

$$\tilde{x} = P_x x, \quad \tilde{u} = P_u u, \quad \tilde{y} = P_y y,$$

где P_x, P_u, P_y матрицы перестановки. Этому соответствует перестановка строк и столбцов матриц A_b, B_b, C_b. Получаем модель $(\tilde{A}, \tilde{B}, \tilde{C})$,

$$\tilde{A} = P_x^T A_b P_x, \quad \tilde{B} = P_x^T B_b P_u, \quad \tilde{C} = P_y^T C P_x.$$

Матрицы подмоделей (A_i, B_i, C_i) получаются выделением из матриц $\tilde{A}, \tilde{B}, \tilde{C}$ блоков на диагонали. Размерности диагональных блоков равняются числу элементов в группах (u_i, x_i, y_i) соответственно. Элементы матриц $\tilde{A}, \tilde{B}, \tilde{C}$ вне диагональных блоков можно на основе предыдущего анализа декомпонируемости приравнять нулю. Получим блочно-диагональные матрицы декомпонируемой модели

$$\tilde{A}_D = \mathrm{diag}(\tilde{A}_i), \quad \tilde{B}_D = \mathrm{diag}(\tilde{B}_i), \quad \tilde{C}_D = \mathrm{diag}(\tilde{C}_i),$$

$$\dim \tilde{A}_i = n_i \times n_i, \quad \dim \tilde{B}_i = n_i \times m_i,$$
$$\dim \tilde{C}_i = r_i \times n_i, \quad 1 \leqslant i \leqslant p,$$
$$n_i = |\mathfrak{X}_i|, \quad m_i = |U_i|, \quad r_i = |Y_i|,$$

где $|\mathfrak{X}_i|$ обозначит число элементов множества $\mathfrak{X}_i$.

Вспомним, что БН модель (A_b, B_b, C_b) была получена нормированием входов и выходов. Для получения декомпонированной модели, где входы и выходы эквивалентны исходной модели, необходимо провести обратное преобразование входов и выходов

$$T_U = (P_U^T \mathfrak{N} P_U)^{-1}, \quad T_Y = (P_Y^T \mathfrak{B} P_Y)^{-1}.$$

Матрицы декомпонированной модели равняются

$$A_D = \tilde{A}_D, \quad B_D = \tilde{B}_D T_U, \quad C_D = T_Y \tilde{C}_D.$$

Относительно взаимосвязи свойств исходной модели и полученных автономных подмоделей справедливы следующие две теоремы [5].

Т е о р е м а 2. Если исходная управляемая и наблюдаемая модель (A, B, C) асимптотически устойчива, то автономные подмодели (A_i, B_i, C_i), $1 \leqslant i \leqslant p$ также асимптотически устойчивы.

Т е о р е м а 3. Автономные подмодели (A_i, B_i, C_i), $1 \leqslant i \leqslant p$ управляемы и наблюдаемы, если исходная модель асимптотически устойчива и в критериях (5) и (6) число ε удовлетворяет условию

$$\varepsilon < \frac{\mu_c}{\sqrt{\max(m,r) \cdot n}},$$

где μ_c - мера управляемости БН модели.

Из теоремы 2 следует, что устойчивость подмоделей гарантирована, если мы исходим из устойчивой управляемой и наблюдаемой модели. Теорема 3 дает достаточное условие управляемости и наблюдаемости автономных подмоделей. В действительности подмодели могут оказаться управляемыми и наблюдаемыми даже при большем значении ε. При неустойчивой исходной модели надо всегда проверить управляемость и наблюдаемость подмоделей. Если подмодели окажутся неуправляемыми или ненаблюдаемыми, надо уменьшить значение ε.

9. Ошибка декомпозиции

Декомпонированная модель получается посредством отбрасывания несущественных связей вход – состояние – выход использованием приведенного алгоритма. Переходные характеристики декомпонированной модели (A_D, B_D, C_D) в некоторой степени отличаются от характеристик исходной модели (A, B, C). Это несоответствие назовем ошибкой декомпозиции. Постараемся ее оценить по импульсным переходным функциям, следуя идеям работы [4].

Допустим, что входы и выходы в исходной и декомпонированной моделях расположены в одинаковой последовательности и одинаково нормированы. Импульсные переходные матрицы исходной и декомпонированной моделей равняются $H(t) = Ce^{At}B$ и $H_D(t) = C_D e^{A_D t} B_D$. В силу приближенности декомпозиции матрицы $H(t)$ и $H_D(t)$ не равны. Импульсной переходной матрицей ошибки назовем матрицу $H_e(t) = H(t) - H_D(t)$. Матрица $H_e(t)$ характеризует ошибку декомпозиции. Декомпозицию можно считать допустимой, если элементы матрицы $H_e(t)$ малые относительно элементов матрицы $H(t)$.

Пусть $\sigma_{max}(W^2_{He})$ максимальная ВГК матрицы $H_e(t)$ и $\sigma_{min}(W^2_H)$ минимальная ВГК матрицы $H(t)$ на промежутке $[0, t_1]$. Тогда относительной ошибкой декомпозиции ε_D назовем отношение

$$\varepsilon_D = \frac{\sigma_{max}(W^2_{He})}{\sigma_{min}(W^2_H)}.$$

Величина $\sigma_{min}(W_H)$ вычисляется посредством сингулярного разложения матрицы $W^2_H = CW^2_e C^T$. Величина $\sigma_{max}(W^2_{He})$ равняется квадратному корню максимального сингулярного значения матрицы

$$W^2_{He} = \int_0^{t_1} H_e H_e^T dt =$$

$$= \int_0^{t_1} Ce^{At} B B^T e^{A^T t} C^T dt - \int_0^{t_1} Ce^{At} B B_D^T e^{A_D^T t} C_D^T dt -$$

$$- \left(\int_0^{t_1} Ce^{At} B B_D^T e^{A_D^T t} C_D^T dt \right)^T + \int_0^{t_1} C_D e^{A_D t} B_D B_D^T e^{A_D^T} C_D^T dt.$$

Если ε_D мала, то импульсные переходные функции ошибки декомпозиции по норме малы относительно импульсных переходных функций объекта. Какое ε_D считать малой, судит проектировщик системы управления.

Литература

I. Яаксоо Ю.И. Исследование взаимосвязей многомерного объекта автоматического управления. - В кн.: Исследования по теории многосвязных систем. М., Наука, 1982, с. 61-65.

2. Андреев Ю.Н. Управление конечномерными линейными объектами. М., Наука, 1976. 424 с.

3. Форсайт Дж., Малькольм М., Моулер К. Машинные методы математических вычислений. М., Мир, 1980. 279 с.

4. Moore B.C. Principal component analysis in linear systems: controllability, observability, and model reduction. - IEEE Trans. Autom. Control, 1981, vol. AC-26, N 1, p. 17-32.

5. Райенд К. Машинное проектирование систем автоматического управления непрерывными процессами. Дис. на соиск. ученой степени канд. техн наук. Таллин, 1983 (Таллинск. политехн. ин-т).

K. Raiend

DECOMPOSITION OF THE LINEAR DYNAMIC SYSTEMS MODEL

Summary

In this paper an algorithm for decomposition of linear dynamic systems model in state space is proposed. Rules to obtain subsystems equations and estimate the decomposition error are given.

№ 550

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED

ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

УДК 65.012.122

А. Кийтам, Э. Сакс

ПРИБЛИЖЕННАЯ ОПТИМИЗАЦИЯ АЛГОРИТМОВ КОМПЕНСАЦИОННОГО УПРАВЛЕНИЯ С РЕЖИМНЫМИ ОГРАНИЧЕНИЯМИ ДЛЯ МНОГОСТАДИЙНЫХ ТЕХНОЛОГИЧЕСКИХ ПРОЦЕССОВ

I. Постановка задачи

Рассматриваем систему компенсационного управления (КУ) для многостадийного технологического процесса (ТП) с групповой обработкой полуфабрикатов (ПФ). Система КУ, функционирующая между двумя последовательными стадиями ТП, предназначена для уменьшения разброса параметров изготовляемых изделий. Система осуществляет распределение поступивших с предыдущей стадии обработки ПФ по достаточно однородным по качеству группам, комплектацию из этих групп партий для последующей обработки и определение режима обработки для скомплектованных партий. В источниках [1, 2] приведены процедуры приближенной оптимизации параметров алгоритма системы КУ для случая, если отсутствуют ограничения при выборе компенсирующего технологического режима. В настоящей статье задача оптимизации параметров алгоритма системы КУ рассматривается для практически важной ситуации, если при выборе компенсирующего технологического режима необходимо учитывать ограничения.

Пусть качество ПФ после завершения предыдущей стадии обработки характеризуется значением скалярной переменной y. Обозначим плотность распределения y через $p_y(y)$ и предположим, что $p_y(y)$ равняется нулю вне конечного отрезка $\alpha \le y \le \beta$. Отрезок $[\alpha, \beta]$ разбивается на m интервалов $[y_{t-1}, y_t]$, $\alpha = y_0 < y_1 < \ldots < y_{m-1} < y_m = \beta$. Поступившие партии ПФ с объемом n_y распределяются в зависимости от значения y по m промежуточным складам Ω_t: при $y_{t-1} < y < y_t$ ПФ

направляется в Ω_t. Если в некотором Ω_t накопилось по крайней мере n_w ПФ, то из них комплектуется новая партия с фиксированным объемом n_w. Эта партия обрабатывается по режиму x_t, который зависит только от t. Целью выбора отличающихся режимов x_t является уменьшение разброса некоторого параметра w, который определяет качество ПФ после последующей обработки и связан с y линейной моделью с гауссовским шумом:

$$z = a + bx + cy, \quad w = z + \eta, \quad \eta \sim N(0, \sigma_\eta), \qquad (1)$$

где a, b, c – постоянные коэффициенты;

σ_η – стандартное отклонение шума η.

Обозначим центр настройки ТП по w через $w_0 = z_0$. Условие бездефектности обработанного ПФ выражается как условие принадлежности w к симметричному относительно w_0 допуску $\varepsilon = w_0 - \delta_w \le w \le w_0 + \delta_w = \lambda$, $\delta_w \ge 0$.

Предположим, что компенсирующие режимы x_t можно выбирать из конечного отрезка $[x_{min}, x_{max}]$, т.е. $x_{min} \le x_t \le x_{max}$. Тогда возможные значения z находятся в конечном отрезке, который определен значениями x_{min}, x_{max}, α и β.

Оптимизируемыми параметрами системы КУ являются:

1) число интервалов m;

2) границы интервалов y_t, $t = \overline{1, m-1}$;

3) режимы обработки x_t, $t = \overline{1, m}$.

2. Процедуры оптимизации

В [1] предложен критерий оптимизации алгоритмов КУ и показано, что при определенных условиях можно задачу определения оптимального числа интервалов m^* сформулировать как задачу определения максимального значения m, удовлетворяющего неравенству

$$\Phi_0\left\{\delta_w / \sqrt{\sigma_\eta^2 + [\sigma_z^2(m)]^*}\right\} - \Phi_0\left\{\delta_w / \sqrt{\sigma_\eta^2 + [\sigma_z^2(m-1)]^*}\right\} > s_0/(4r_0) = d, \qquad (2)$$

где $\Phi_0(\tau)$ – функция распределения нормированного гауссовского распределения $N(0,1)$:

$$\Phi_0(\tau)=\int_{-\infty}^{\tau}\varphi(u)\,du,\quad \varphi_0(u)=\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{u^2}{2}\right). \qquad (3)$$

s_0 – затраты на хранение одного ПФ в течение цикла производства;

r_0 – затраты, возникшие из-за появления дефектного по w ПФ;

$[\sigma_z^2(m)]^*$ – минимальное достигаемое значение дисперсии z относительно центра настройки z_0 при числе интервалов m.

Минимальное значение $\sigma_z^2(m)$ при каждом m достигается оптимальным выбором границ интервалов y_t и режимов обработки x_t. В отличие от [1, 2] выбор x_t в рассматриваемом случае должен подчиняться ограничениям $x_{min}\le x_t\le x_{max}$; это приводит к необходимости модификации процедур вычисления оптимальных y_t, x_t и $[\sigma_t^2(m)]^*$ при фиксированном значении m.

Таким образом, требуется путем выбора границ интервалов y_t и режимов обработки x_t, $x_{min}\le x_t\le x_{max}$ минимизировать $\sigma_z^2(m)$:

$$\sigma_z^2(m)=\int_{-\infty}^{\infty}(z-z_0)^2 p_z(z)\,dz\to\min.$$

Отсюда для модели (1) получим экстремальную задачу

$$\sigma_z^2(m)=\sum_{t=1}^{m}\int_{y_{t-1}}^{y_t}(a+bx_t+cy-z_0)^2 p_y(y)\,dy\to\min, \qquad (4)$$
$$x_{min}<x_t\le x_{max},\quad t=\overline{1,m}.$$

Осуществляя в (4) замену переменных

$$v_t=\frac{1}{c}(z_0-a-bx_t),\ t=\overline{1,m}, \qquad (5)$$

получим

$$\sigma_z^2(m)=c^2D_y=c^2\sum_{t=1}^{m}I_t\to\min,\ I_t=\int_{y_{t-1}}^{y_t}(y-v_t)^2 p_y(y)\,dy, \qquad (6)$$
$$y_{min}\le v_t\le y_{max},\quad t=\overline{1,m},$$

где y_{min} является минимальным, а y_{max} максимальным значением y, для которого существует такое x, что $x\in[x_{min}, x_{max}]$ при $z_0=a+bx+cy$:

$$y_{min}=y_\infty-\left|\frac{b}{c}\right|\delta_x,\quad y_{max}=y_\infty+\left|\frac{b}{c}\right|\delta_x,$$

$$y_\infty = \frac{z_0 - a}{c} - \frac{b}{c} \cdot \frac{x_{min} + x_{max}}{2}; \qquad \sigma_x = \frac{x_{max} - x_{min}}{2}. \qquad (7)$$

Из (6) видно, что для достижения минимума $\sigma_z^2(m)$ требуется $y_{t-1} < v_t < y_t$, $t = \overline{1,m}$, откуда получим $v_1 < v_2 < \dots < v_{m-1} < v_m$. С учетом этого в (6) достаточно учитывать два ограничения:

$$v_1 \geq y_{min}, \quad v_m \leq y_{max}. \qquad (8)$$

Величины α, β, y_{min}, y_{max}, характеризующие систему КУ, определяют существенность ограничения $x_{min} < x \leq x_{max}$. Если $[\alpha, \beta] \subset [y_{min}, y_{max}]$, то ограничение $x \in [x_{min}, x_{max}]$ не является существенным и можно применять процедуры, приведенные в [1, 2].

Далее рассмотрим ограниченную с обоих сторон систему КУ ($\alpha < y_{min}, y_{max} < \beta$), как наиболее общий случай. Согласно (6), оптимальные v_t при фиксированных границах интервалов y_t определяются в результате минимизации I_t. Решая уравнения $\partial I_t / \partial v_t = 0$, получаем для средних классов

$$v_t = \frac{\int_{y_{t-1}}^{y_t} y\, p_y(y)\, dy}{\pi_t = \int_{y_{t-1}}^{y_t} p_y(y)\, dy} = \mu_t(y), \quad t = \overline{2, m-1}, \qquad (9)$$

где π_t - является вероятностью, а

$\mu_t(y)$ - средним значением для t-го интервала.

При минимизации I_1, I_m следует учитывать ограничение (8). Используя приведенные в работах [3, 4] процедуры нелинейного программирования, получим четыре возможных оптимальных варианта v_1, v_m:

$$\begin{aligned}
&1.\ v_1^{(1)} = \mu_1(y); \quad v_m^{(1)} = \mu_m(y) \\
&2.\ v_1^{(2)} = \mu_1(y); \quad v_m^{(2)} = y_{max} \\
&3.\ v_1^{(3)} = y_{min}; \quad v_m^{(3)} = \mu_m(y) \\
&4.\ v_1^{(4)} = y_{min}; \quad v_m^{(4)} = y_{max}.
\end{aligned} \qquad (10)$$

Здесь и в дальнейшем пронумеруем возможные комбинации от I до 4, а соответствующие i-й комбинации величины обозначим через верхний индекс i в круглых скобках.

Подставляя (10) и (9) в (6), получим следующие критерии определения оптимальных границ:

$$\text{I. } D_y^{(1)} = \sum_{t=1}^{m} \pi_t \cdot \sigma_t^2(y) \to \min$$

$$\text{2. } D_y^{(2)} = \sum_{t=1}^{m-1} \pi_t \cdot \sigma_t^2(y) + \int_{y_{m-1}}^{\beta} (y - y_{max})^2 p_y(y)\, dy \to \min$$

$$\text{3. } D_y^{(3)} = \sum_{t=2}^{m} \pi_t \cdot \sigma_t^2(y) + \int_{\alpha}^{y_1} (y - y_{min})^2 p_y(y)\, dy \to \min \qquad \text{(II)}$$

$$\text{4. } D_y^{(4)} = \sum_{t=2}^{m-1} \pi_t \cdot \sigma_t^2(y) + \int_{\alpha}^{y_1} (y - y_{min})^2 p_y(y)\, dy + \int_{y_{m-1}}^{\beta} (y - y_{max})^2 p_y(y)\, dy \to \min,$$

где $\sigma_t^2(y)$ - дисперсия t-го интервала: $\sigma_t^2(y) = \int_{y_{t-1}}^{y_t} (y - \mu_t(y))^2 p_y(y)\, dy / \pi_t$.

Из (II) видно, что определение y_t путем непосредственной минимизации $D_y^{(i)}$ или решением системы уравнений $\partial D_y^{(i)} / \partial y_t = 0$, $t = \overline{1,m-1}$ приводит к явным аналитическим формулам лишь в некоторых частных случаях; для большинства практических распределений $p_y(y)$ необходимо привлечение численных методов расчета y_t. Поэтому целесообразно поставить вопрос об отыскании простой приближенной процедуры определения y_t. Для конструирования такой процедуры применим следующую доказанную в [2] теорему о среднем значении.

Теорема I. Если $p_y(y)$ непрерывна на $[\alpha,\beta]$, то для любого интервала $[y_{t-1}, y_t]$ существует такая внутренняя точка интервала θ_t, $y_{t-1} < \theta_t < y_t$, что $\pi_t \cdot \sigma_t^2(y) = p_y(\theta_t) \cdot (y_t - y_{t-1})^3 / 12$. Приближенное решение для варианта $i = I$ (т.е. в (II) $D_y^{(1)} \to \min$) найдено в работе [2]. Оптимальные $y_t^{(1)}$ выражаются в виде

$$y_t^{(1)} = \psi_o^{-1}\left[\frac{t}{m}\psi_o(\beta)\right], \qquad \text{(I2)}$$

где здесь и в дальнейшем $\psi_o(y) = \int_{\alpha}^{y} \sqrt{p_y(\tau)}\, d\tau$ и $\psi_o^{-1}(\cdot)$ обозначит обратную функцию для функции $\psi_o(\cdot)$. Так как варианты $i = 2$, $i = 3$ в (II) являются частными случаями варианта $i = 4$, можно ограничиваться рассмотрением варианта $i = 4$. Таким образом, рассмотрим задачу минимидации $D_y^{(4)}$ далее в (II).

Обозначая $p_y(\theta_t) = p_t$, $y_t - y_{t-1} = h_t$, $H = \beta - \alpha$ и используя теорему I, получаем из (II) экстремальную задачу:

$$D_y^{(4)} = \frac{1}{12}\sum_{t=1}^{m} p_t \cdot h_t^3 + \int_{\alpha}^{h_1+\alpha} (y - y_{min})^2 p_y(y)\,dy + \int_{\beta-h_m}^{\beta} (y - y_{max})^2 \cdot p_y(y)\,dy \to \min$$

с ограничениями $\sum_{t=1}^{m} h_t = H$, $h_t > 0$. Образуем лагранжиан

$$\Lambda = D_y^{(4)} - \lambda \left(\sum_{t=1}^{m} h_t - H \right).$$

Система уравнений для определения h_t имеет вид:

$$\partial\Lambda/\partial h_1 = (y_1 - y_{min})^2 p_y(y_1) - \lambda = 0,$$
$$\partial\Lambda/\partial h_t = p_t \cdot h_t^2/4 - \lambda = 0, \quad t = \overline{2,m-1},$$
$$\partial\Lambda/\partial h_m = (y_{m-1} - y_{max})^2 \cdot p_y(y_{m-1}) - \lambda = 0.$$

Отсюда получаем:

$$\begin{aligned} &\sqrt{p_y(y)} \cdot (y_1 - y_{min}) = \sqrt{\lambda}, \\ &\sqrt{p_t} \cdot h_t = 2\sqrt{\lambda}, \ t = \overline{2,m-1}, \\ &\sqrt{p_y(y_{m-1})} \cdot (y_{max} - y_{m-1}) = \sqrt{\lambda}. \end{aligned} \qquad (I3)$$

Из (I3) видно, что оптимальные границы $y_t^{(4)}$, $t = \overline{1,m-1}$ не зависят от α, β и расположены все внутри интервала $[y_{min}, y_{max}]$.

Далее, так как при малых h_t значения $\sqrt{p_t} = \sqrt{p_y(\theta_t)}$, $y_{t-1} < \theta_t < y_t$ можно считать приблизительно равными средним значениям $\sqrt{p_y(y)}$ на интервалах $[y_{t-1}, y_t]$, т.е. $\sqrt{p_t} = \int_{y_{t-1}}^{y_t} \sqrt{p_y(y)}\,dy / h_t$ и аналогично можно $\sqrt{p_y(y_1)}$ и $\sqrt{p_y(y_{m-1})}$ заменять средними значениями $\sqrt{p_y(y)}$ на отрезках $[y_{min}, y_1]$ и $[y_{m-1}, y_{max}]$, получаем для определения y_t из (I3) уравнение

$$\int_{y_{min}}^{y_t} \sqrt{p_y(y)}\,dy = \frac{2t-1}{2(m-1)} \int_{y_{min}}^{y_{max}} \sqrt{p_y(y)}\,dy.$$

Соответственно, формула для приближенного расчета y_t при $\nu = 4$ выражается в виде

$$y_t^{(4)} = \psi_1^{-1}\left[\frac{2t-1}{2(m-1)}\,\psi_1(y_{max})\right]; \qquad (I4)$$

в (I4) и в дальнейшем $\psi_1(y) = \int_{y_{min}}^{y} \sqrt{p_y(\tau)}\, d\tau$.

Как отмечалось, решение задач $D_y^{(2)} \to \min$, $D_y^{(3)} \to \min$ для вариантов $i = 2, 3$ аналогично решению задач $D_y^{(4)} \to \min$ для $i = 4$. Поэтому приведем здесь лишь найденные формулы для приближенного расчета y_t:

$$y_t^{(2)} = \psi_0^{-1}\left[\frac{2t}{2m-1}\psi_0(y_{max})\right] \qquad (I5)$$

$$y_t^{(3)} = \psi_1^{-1}\left[\frac{2t-1}{2m-1}\psi_1(\beta)\right]. \qquad (I6)$$

Таким образом, формулами (I2), (I4), (I5), (I6) определены y_t, $t = \overline{1, m-1}$ для всех $i = \overline{1,4}$. Подставляя полученные $y_t^{(i)}$ в формулу (I0), найдем значения для $v_1^{(i)}$, $v_m^{(i)}$.

Рассматриваемый i-й вариант не является оптимальным, в том случае, если $v_1^{(i)}$, $v_m^{(i)}$ не удовлетворяют ограничениям (8).

Рассмотрение вариантов следует начать с $i = 1$ и провести в возрастающем по i порядке, так как по (II) всегда выполнено $D_y^{(1)} < D_y^{(2)}$, $D_y^{(3)} < D_y^{(4)}$.

Ход нахождения оптимального решения можно представить в виде следующего алгоритма.

Алгоритм I.

I. Найти по (I2), (9), (I0) $y_t^{(1)}$, $v_t^{(1)}$.

Если $v_1^{(1)} \geq y_{min}$, $v_m^{(1)} \leq y_{max}$, то оптимальное решение найдено: $i^* = 1$, $y_t^* = y_t^{(1)}$, $v_t^* = v_t^{(1)}$.

Если $v_1^{(1)} < y_{min}$, $v_m^{(1)} > y_{max}$, то решением является $i = 4$. Переход к п. 4.

Если $v_1^{(1)} < y_{min}$, $v_m^{(1)} \leq y_{max}$, то переход к п. 3.

Если $v_1^{(1)} \geq y_{min}$, $v_m^{(1)} > y_{max}$, то переход к п. 2.

2. Найти по (I5), (9), (I0) $y_t^{(2)}$, $v_t^{(2)}$.

Если $v_1^{(2)} \geq y_{min}$, то оптимальное решение найдено $i^* = 2$, $y_t^* = y_t^{(2)}$, $v_t^* = v_t^{(2)}$.

Если $v_1^{2} < y_{min}$, то решением является $i = 4$. Переход к п. 4.

3. Найти по (I6), (9), (I0) $y_t^{(3)}, v_t^{(3)}$.

Если $v_m^{(3)} \le y_{max}$, то оптимальное решение найдено $i^* = 3$, $y_t^* = y_t^{(3)}$, $v_t^* = v_t^{(3)}$.

Если $v_m^{(3)} > y_{max}$, то решением является $i = 4$. Переход к п. 4

4. Найти по (I4), (9), (I0) $y_t^{(4)}, v_t^{(4)}$; $i^* = 4$, $y_t^* = y_t^{(4)}$, $v_t^* = v_t^{(4)}$.

Подставляя найденные i^*, y_t^*, v_t^* в (II), получаем достигаемое при числе интервалов m минимальное значение $[\sigma_z^2(m)]^* = c^2 D_y^*$.

При вычислении оптимальной m^* по (2) требуется найти $[\sigma_z^2(m)]^*$ для текущих значений m, $m = \overline{2, m^*+1}$. Объем вычислений по алгоритму I для $m = \overline{2, m^*+1}$ можно значительно уменьшить, если учитывать, что $v_1^{(i)}(k), v_k^{(i)}(k)$ при числе интервалов $k = m, m+1$ всегда удовлетворяют неравенствам $v_1^{(i)}(m+1) \le v_1^{(i)}(m)$; $v_{m+1}^{(i)}(m+1) \ge v_m^{(i)}(m)$, и при числе интервалов $m+1$ следует выполнить алгоритм I не с пункта I, а с пункта $i^*(m)$, где $i^*(m)$ – номер найденного при числе интервалов m оптимального варианта.

Выводимые при предположении $[y_{min}, y_{max}] \subset [\alpha, \beta]$ процедуры оптимизации непосредственно применимы и для неограниченных или ограниченных с одной стороны систем КУ. Отметим следующее: I) для неограниченных систем $i^*(m) = 1$ независимо от m; 2) для ограниченных с одной стороны систем существует такое значение m_ℓ, что при $m < m_\ell$ $i^*(m) = 1$ и при $m \ge m_\ell$ $i^*(m) = 2$ или 3; 3) для ограниченных с обеих сторон систем существует дополнительно такое m_k, $m_k \ge m_\ell$, что $i^*(m) = 4$ при $m > m_k$.

Полученные алгоритмы оптимизации КУ прямо обобщаемы на систему КУ с векторными входами $\mathfrak{X}, Y$ и с ограниченной областью Ω_x вариации режимов обработки $\mathfrak{X}$, если эти системы КУ можно описать моделью в виде

$$w = a + F(\mathfrak{X}) + \sum_j c_j \cdot g_j(Y) + \eta,$$

где a, c_j – коэффициенты модели.

Приняв в качестве x, y, в формуле (I) $x = F(\mathfrak{X})$ и

$y = \sum_j c_j g_j(Y)$ непосредственно можем применить приведенные процедуры.

3. Пример

Иллюстрируем применение приведенных процедур на примере гауссовского распределения y. Пусть $p_y(y)$ образуется в результате усечения нормированного гауссовского распределения $N(0,1)$ на отрезке $[\alpha, \beta]$:

$$p_y(y) = \frac{1}{u} \cdot \varphi_0(y), \quad u = \Phi_0(\beta) - \Phi_0(\alpha). \qquad (I7)$$

Отмечаем, что среднее значение μ_y и дисперсия σ_y^2 выражаются для (I7) в виде:

$$\mu_y = [\varphi_0(\alpha) - \varphi_0(\beta)] / u$$
$$\sigma_y^2 = 1 + [\alpha \cdot \varphi_0(\alpha) - \beta \cdot \varphi_0(\beta)] / u - [\mu_y]^2.$$

Подставляя (I7) в (I2), (I4)÷(I6), получаем после некоторых преобразований для вычисления $y_t^{(i)}$ формулу

$$y_t^{(i)} = \sqrt{2}\, \Phi_0^{-1}\left\{\Phi_0\left(\frac{\gamma_i}{\sqrt{2}}\right) + \kappa_i\left[\Phi_0\left(\frac{\eta_i}{\sqrt{2}}\right) - \Phi_0\left(\frac{\gamma_i}{\sqrt{2}}\right)\right]\right\}, \quad i = \overline{1,4},$$

где $\gamma_1 = \gamma_2 = \alpha$; $\gamma_3 = \gamma_4 = y_{min}$; $\eta_1 = \eta_3 = \beta$; $\eta_2 = \eta_4 = y_{max}$;
$\kappa_1 = t/m$; $\kappa_2 = 2t/(2m-1)$; $\kappa_3 = (2t-1)/(2m-1)$; $\kappa_4 = (2t-1)/(2m-2)$.

Формулы (II) для $D_y^{(i)}$ можно представлять в более удобной в вычислительном смысле форме:

$$D_y^{(i)} = \sigma_y^2 + \mu_y^2 - \left\{\sum_{t=1}^{m} \pi_t [\mu_t(y)]^2\right\} / u + s_i, \quad i = \overline{1,4},$$
$$s_1 = 0; \quad s_2 = \tau_1; \quad s_3 = \tau_2; \quad s_4 = \tau_1 + \tau_2;$$
$$\tau_1 = \pi_m \cdot [\mu_m(y) - y_{max}]^2 / u;$$
$$\tau_2 = \pi_1 \cdot [\mu_1(y) - y_{min}]^2 / u;$$
$$\mu_t(y) = [\varphi_0(y_{t-1}) - \varphi_0(y_t)] / \pi_t;$$
$$\pi_t = \Phi_0(y_t) - \Phi_0(y_{t-1}).$$

Для иллюстрации хода вычислений по приведенным процедурам рассмотрим ограниченную с обеих сторон систему КУ с $y_{min} = -1{,}5$, $y_{max} = 2{,}0$, при неусеченном гауссовским входным распределением $N(0,1)$ с $\alpha = -\infty$, $\beta = \infty$.

В таблице I приведены результаты оптимизации СКУ. Для каждой m приведены номера i рассмотренных в алгоритме I вариантов, полученные $v_1^{(i)}$, $v_m^{(i)}$ и оптимальное D_y. Для

оценки влияния ограничений на эффективность КУ в последнем столбце таблицы приведены значения D_y для неограниченной системы КУ. С "х" в таблице обозначены значения $v_1^{(i)}$, $v_m^{(i)}$, не удовлетворяющие ограничениям (8). Отметим, что $m_\ell = 5$ $m_к = 8$. Для значений $m < m_\ell$ D_y ограниченной и неограниченной системы совпадают, для $m \geq m_\ell$ D_y уменьшается с увеличением m значительно медленнее, чем D_y°.

Таблица I

Пример оптимизации СКУ

m	i	$v_1^{(i)}$	$v_m^{(i)}$	D_y	D_y°
2	I	-0,798	0,798	0,363	0,363
3	I	-I,2I9	+I,2I9	0,I90	0,I90
4	I	-I,483	I,483	0,II8	0,II8
5	I	-I,673^x	I,673		
	3	-I,500	I,7I3	0,083	0,080
6	3	-I,500	-I,87I	0,064	0,058
7	3	-I,500	I,996	0,053	0,045
8	3	-I,500	2,I00^x		
	4	-I,500	2,000	0,047	0,035
9	4	-I,500	2,000	0,042	0,029
I0	4	-I,500	2,000	0,039	0,024

Литература

I. Кийтам А.А. Приближенная оптимизация алгоритмов компенсационного управления для многостадийных технологических процессов I. - Автоматика и телемеханика, I98I, № 8, с. I72-I78.

2. Кийтам А.А. Приближенная оптимизация алгоритмов компенсационного управления для многостадийных технологических процессов II. - Автоматика и телемеханика, I98I, № I0, с, I77-I82.

3. Зангвилл У.И. Нелинейное программирование. М., Сов. Радио, I973.

4. Хедли Дж. Нелинейное и динамическое программирование. М., Мир, I967.

A. Kiitam, E. Saks

APPROXIMATE OPTIMIZATION OF COMPENSATIONAL CONTROL ALGORITHMS WITH REGIME LIMITATIONS FOR MULTI-STAGE TECHNOLOGICAL PROCESSES

Summary

The paper is concerned with a compensational control system with regime limitations for a multi-stage technological process which is described by a linear model with Gaussian noise. Optimization procedure is presented, particularly for the criterion of output variance minimization. Some generalizations are mentioned. An example is given.

№ 550

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED

ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

УДК 681.142.2

Е. Венделин

ИДЕНТИФИКАЦИЯ ОБЪЕКТОВ ПРИ ПЛАНИРОВАНИИ ВЫЧИСЛЕНИЙ

I. Введение

Идея создания пакета программ на базе какой-либо инструментальной системы получает все более широкое распространение. Такая система обеспечивает выполнение функций, не зависящих от конкретной предметной области, и позволяет автоматизировать все основные этапы разработки пакета: создание входного языка, составление вычислительных алгоритмов, вывод результатов. Тем самым эта система позволяет разработчику пакета работать на содержательном уровне.

Разработчик составляет описание предметной области, т.е. объектов данных и связывающих их отношений в виде какой-либо модели. Описывая задачу, пользователь присваивает одним объектам из модели значения, а другие указывает как искомые, и на основе этой информации системой частично или полностью планируется вычислительный процесс [1, 2].

Система SPADE является инструментальной системой, ориентированной, в основном, на разработку пакетов проектирования электронных схем и систем управления. Основной особенностью этих предметных областей является то, что они представляют собой некоторый набор связанных объектов, причем эти связи могут быть различными в различных конкретных задачах. Итак, исходные данные задачи содержат не только значения объектов, но и описание структуры, в которую они соединены. В модели предметной области описывается набор типов объектов, на основе которых задаются конкретные экземпляры объектов для решения задачи. Например, описав предметную область "электрическая цепь" с объектами "резистор" и "конденсатор", можно задавать любые схемы, содержащие эти элементы [3, 4].

Описываемые в модели отношения связывают атрибуты определенных типов объектов, а в конкретной задаче они должны быть привязаны к каждому экземпляру соответствующего типа. Возникает вопрос, какие экземпляры объектов связать между собой тем или иным отношением и один из методов его решения является темой настоящей статьи.

2. Модель предметной области

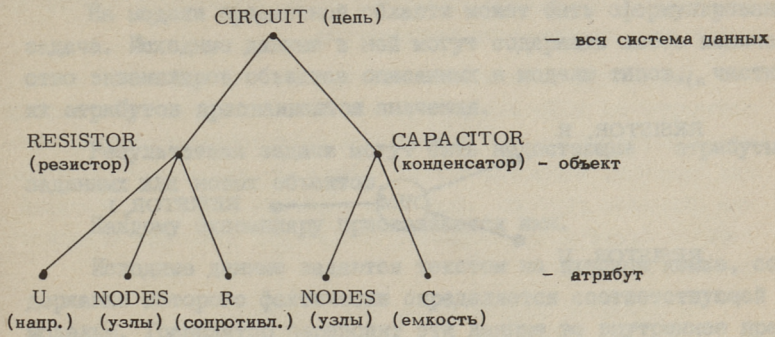
Модель предметной области в системе SPADE как на логическом, так и на физическом уровне делится на две составные части: описание данных и описание отношений.

Для описания данных выбрана иерархическая структура, в которой выделены три основных уровня: уровень всей системы данных, уровень объекта и уровень атрибута. На уровне объекта описываются все типы объектов, используемые в предметной области (например, типы элементов схемы, функций и т.д.). Характеристики объектов содержатся в атрибутах и каждому типу объекта соответствует определенный их набор (например,номера узлов и сопротивление у резистора, нули и полюсы у передаточной функции). В конкретной задаче атрибуту объекта соответствует вектор значений, а в модели предметной области задается некоторая информация об этих значениях, такая, как их тип, количество, область определения и т.д. Элементы каждого уровня структуры снабжены идентификаторами, которые при необходимости могут быть использованы в качестве ссылки на какой-либо атрибут (фиг. I а).

Отношение в модели определяет принципиальную возможность вычисления значений атрибутов какого-либо типа объекта из значений атрибутов объекта такого же или другого типа.Его описание состоит из имени оператора (способа вычисления) и ссылок на данные (атрибуты объектов) (фиг. 2 а). В системе SPADE отношение может иметь любое количество входов и один выход.

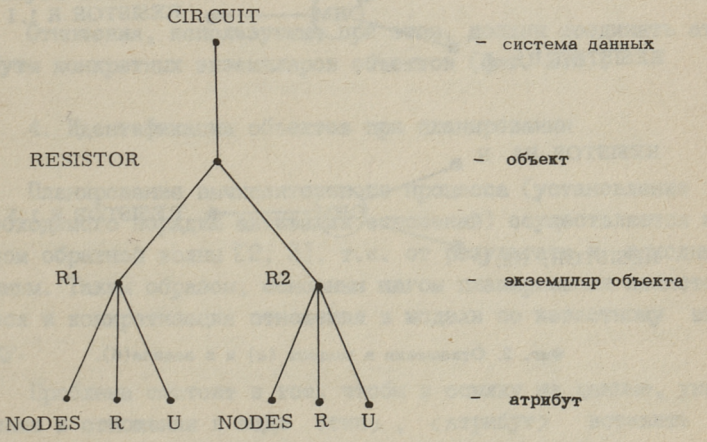
Модель предметной области хранится в архиве и вызывается из него при решении соответствующей данной модели задачи.

а)



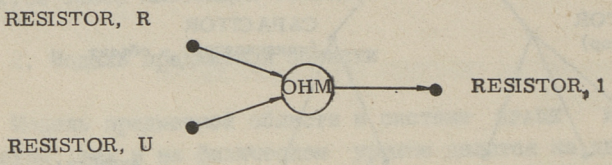
б)

```
CIRCUIT DESCRIPTION
R1 RESISTOR NODES = 1, 2 R = 1. U = 1.;
R2 RESISTOR NODES = 3, 4 R = 1. U = 0, 5.;
```

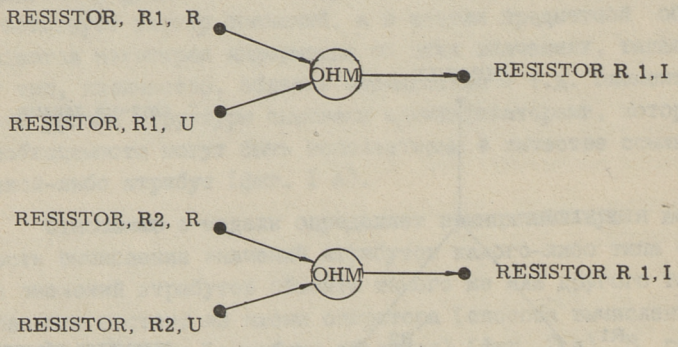


Фиг. 1. Описание данных в модели (а) и в задаче (б).

а)



б)



Фиг. 2. Отношения в модели (а) и в задаче(б).

3. Данные и отношения в задаче

На модели предметной области может быть сформулирована задача. Исходные данные в ней могут содержать любое количество экземпляров объектов описанных в модели типов, части их атрибутов присваиваются значения.

Результатами задачи могут быть недостающие атрибуты заданных или новых объектов.

Каждому экземпляру присваивается имя.

Исходные данные задаются текстом на входном языке, содержание которого фактически определяется соответствующей моделью. Транслятор переводит эти данные во внутреннее представление, структуру данных задачи, проверяя их соответствие описанию данных модели. Структура данных имеет четыре основных уровня: уровень системы данных, уровень объекта, уровень экземпляра объекта и уровень атрибута (фиг. I.б).

Вычислительный процесс представляет собой постепенное пополнение структуры данных задачи до тех пор, пока атрибуты, указанные как результаты, не будут в нее включены. Каждый шаг процесса - это активация одного отношения, в результате чего в структуру добавляется вектор значений одного атрибута [3].

Отношения, используемые при этом, должны соединять атрибуты конкретных экземпляров объектов (фиг. 2.б).

4. Идентификация объектов при планировании

Планирование вычислительного процесса (установление необходимого порядка активации отношений) осуществляется методом обратной волны [2, 5], т.е. от результата к исходным данным. Таким образом, основным шагом планирования является поиск и конкретизация отношения в модели по известному выходу.

Проблема состоит в том, чтобы в ссылку на данные, указанные в отношении в виде <тип>, <атрибут> вставить правильное имя экземпляра объекта и получить ссылку <тип>, <имя>, <атрибут>.

Учитывая то, что в данных задачи может быть описано несколько однотипных объектов, а в модели несколько отношений для вычисления одного атрибута, часто проведение такого согласования оказывается невозможным.

Например, имея в качестве предметной области схему, состоящую из цифровых фильтров, мы хотим определить задачу, где в исходных данных задано два последовательно соединенных фильтра. Выходной сигнал одного из них является входным сигналом другого. Известны значения входного сигнала первого фильтра, а вычислить необходимо значения выходного сигнала второго.

В описании задачи имеем три сигнала:

```
S1  SIGNAL  FUNS = <значения>;
S2  SIGNAL;
S3  SIGNAL;
```

В модели описано отношение FILT, вычисляющее из значений одного сигнала значения другого, но понятно, что один из них должен быть входным, а другой - выходным сигналом фильтра. Поэтому приходится вводить уточнения.

Определим объект типа FILTER и опишем нашу задачу в виде:

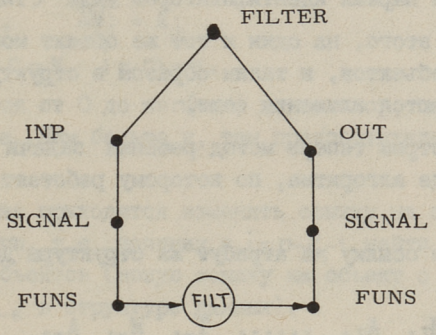
```
H1  FILTER  INP = S1  OUT = S2;
H2  FILTER  INP = S2  OUT = S3;

S1  SIGNAL FUNS =    <значения>;
S2  SIGNAL;
S3  SIGNAL;
```

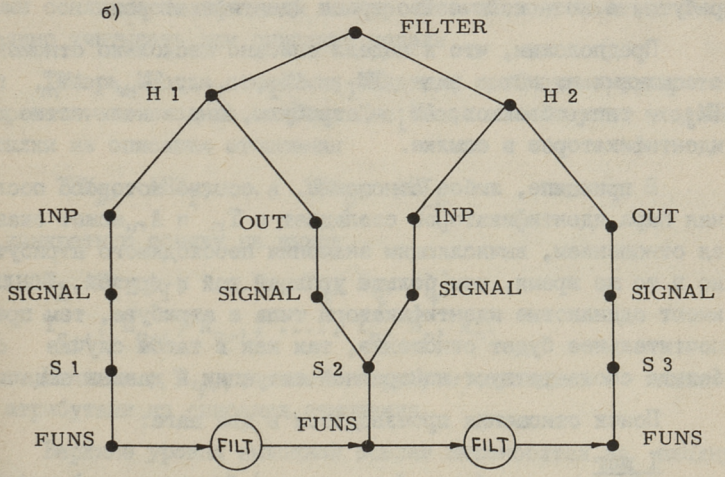
Таким образом, мы определили новый уровень в структуре данных задачи путем введения ссылки с одного объекта на другой. В тексте задания эта ссылка выражается в том, что атрибут одного объекта имеет своим значением имя другого объекта. Ссылка на атрибуты нижнего уровня может теперь состоять из нескольких идентификаторов типа, имени и атрибута (см. фиг. 3).

Глубина структуры данных не ограничена, но ссылка на атрибут самого нижнего уровня всегда образуется одной или несколькими группами идентификаторов <тип> , <имя> , <атрибут> .

а)



б)



Фиг. 3. Описание сложных обьектов в модели (а) и в задаче (б).

Кроме этого всегда сохраняется и прямой доступ к объекту. В описании данных модели в информации о значениях атрибутов, через которые можно ссылаться на другие объекты, указываются допустимые типы подчиненных объектов, а в описании отношения ссылка на данные может быть образована одной или несколькими парами идентификаторов вида <тип> , <атрибут> .

Кроме этого, на один и тот же объект можно ссылаться из разных объектов, и таким образом в структуре данных задачи появляются элементы сети.

Рассмотрим теперь метод решения задачи о выборе отношения в виде алгоритма, по которому работает планировщик системы.

Опишем ссылку на атрибут из структуры данных задачи в виде

$$T_I,\ И_I,\ A_I,\ \ldots\ldots\ T_n,\ И_n,\ A_n,$$

где T_i - типы объектов, $И_i$ - имена экземпляров, A_i - атрибуты, а n - количество троек идентификаторов.

Предположим, что в модели описано несколько отношений с ссылками на выход вида TM_I, AM_I, ..., TM_m, AM_m, где TM_j - типы объектов, AM_j - атрибуты, а m - количество пар идентификаторов в ссылке.

В принципе, любое отношение, в ссылке которого последняя пара идентификаторов совпадает с T_n и A_n, может оказаться отношением, вычисляющим значения необходимого атрибута, но в то же время, чем больше уровней той и другой ссылки имеют одинаковые идентификаторы типа и атрибута, тем предпочтительнее будет отношение, так как в таком случае оно больше соответствует конкретной ситуации в данных задачи.

Поиск отношения производится в два шага:

I шаг

Из отношений в модели выбираются те, у которых в ссылке на выход

1) $m \leqslant n$

2) $TM_j = T_i,\ AM_j = A_i$
для $i = n-m+1, \ldots, n,\ j = 1, \ldots, m.$

Если такие отношения существуют, из них выбирается то, у которого разность $n-m$ наименьшая, и поиск заканчивается. Если же их нет, переходят ко второму шагу.

2 шаг

Выбираем отношения, где

$TM_j = T_i$, $AM_j = A_i$

для $i = n-k$ $j = m-k$,

а к изменяется от 0 до $\min(m,n)$ до тех пор, пока соблюдаются равенства. Чем больше к, тем предпочтительнее отношение.

Теперь уже приходится изменять ссылку на заданный атрибут из задачи, т.е., начиная с $j = n-k-1$, найти новые имена экземпляров объектов (новую ссылку на объект с типом T_{n-k} и именем $И_{n-k}$ в структуре данных).

Отношение нельзя однозначно определить в одном случае - тогда, когда в модели существуют два отношения с одинаковыми ссылками на выходной вектор. Это обстоятельство необходимо учитывать при описании модели.

Теперь, когда отношение найдено, необходимо конкретизировать его входные ссылки. Для этого используют входные ссылки из описания отношения

$$ТВХ_I,\ АВХ_I,\ \ldots,\ ТВХ_p,\ АВХ_p$$

и конкретную ссылку на выход

$$T_I, И_I, A_I, \ldots\ldots, T_q, И_q, A_q;$$
$$T_{q+1}, И_{q+1}, A_{q+1}, \ldots., T_n, И_n, A_n.$$

Начиная с T_{q+1}, типы и атрибуты совпадают с типами и атрибутами из описания отношения.

Верхние уровни выходной ссылки переносятся на входную ссылку без изменений (уровни от I до q), а имена объектов на нижних уровнях образуются по следующему алгоритму:

I) $i := q+1$,

2) если $ТВХ_i = T_i$, то $ИВХ_i = И_i$,
в противном случае идти к 5,

3) если $АВХ_i \neq A_i$, идти к 5,

4) если $i < p$ то $i := i+1$, идти к 2,

5) конец.

В результате будет получена входная ссылка в виде:

$ТВХ_I$, $ИВХ_I$, $АВХ_I$, ..., $ТВХ_q$, $ИВХ_q$, $АВХ_q$;

$ТВХ_{q+1}$, $ИВХ_{q+1}$, $АВХ_{q+1}$, ..., $ТВХ_i$, $ИВХ_i$, $АВХ_i$;

$ТВХ_{i+1}$, $ИВХ_{i+1}$, $АВХ_{i+1}$, ..., $ТВХ_l$, $ИВХ_l$, $АВХ_l$.

Здесь уровни от I до q перенесены из выходной ссылки, типы и атрибуты на уровнях от $q+1$ до l взяты из описания отношения ($l-q=p$). Имена объектов на уровнях от $q+1$ до i взяты из выходной ссылки по приведенному выше алгоритму.

Остальные имена (на уровнях от $i+1$ до l) оставались неизвестными. Начиная с $i+1$-го уровня делается попытка найти как можно больше имен из структуры данных, недостающие имена генерируются.

После того, как все имена объектов во входной ссылке определились, в структуре данных создается вектор, соответствующий этой ссылке, исключая лишь нижний уровень значений атрибута, которые должны появиться в ходе вычислений.

Описанный выше подход накладывает на описание модели два ограничения.

Во-первых, ссылки на выход и входы отношения должны начинаться с одного типа объекта.

Во-вторых, атрибуты промежуточных уровней могут иметь значениями только по одному объекту одного и того же типа.

Более подробно эти алгоритмы будут рассмотрены на примере задачи на схеме, состоящей из цифровых фильтров.

5. Пример

Будем считать, что в описании данных модели имеем объекты:

FILTER - атрибуты INP, OUT, TF (вход, выход, передаточная функция)

SIGNAL - атрибут FUNS,

TF - атрибут COEFF,

SEQ - атрибут FUN,

SIN - атрибуты AMPL, FREQ

Атрибуты INP и OUT могут ссылаться на объекты типа SIGNAL, атрибут FUNS - на объекты типа SEQ и SIN, а атрибут TF - на объект типа TF.

Комплект отношений может быть следующим:

FILT - входы 1) FILTER, INP, SIGNAL, FUNS, SEQ, FUN

2) FILTER, TF, TF, COEFF

выход FILTER, OUT, SIGNAL, FUNS, SEQ, FUN

SEQM - входы 1) SIGNAL, FUNS, SIN, AMPL

2) SIGNAL, FUNS, SIN, FREQ

выход SIGNAL, FUNS, SEQ, FUN

Исходные данные задачи следующие:

```
H1  FILTER  INP = S1  OUT = S2  TF = T1;
H2  FILTER  INP = S2  OUT = S3  TF = T2;
S1  SIGNAL  FUNS= SIN1;
S2  SIGNAL;
S3  SIGNAL; FUNS= SEQ3;
SEQ3  SEQ;
T1  TF  COEFF = ...; T2  TF  COEFF = ...;
SIN1  SIN  AMPL = ...  FREQ = ...;
```

Предположим, что хотим найти выходной сигнал фильтра H2 (FILTER, H2, OUT, SIGNAL, S3, FUNS, SEQ, SEQ3, FUN)

Планировщик начинает поиск пути с результатов в следующем порядке.

I. Выбирается отношение FILT, так как в его выходной ссылке с ссылкой на результат совпадают типы и атрибуты на трех уровнях, а в отношении SEQM только на двух.

2. Формируются входные ссылки

1) FILTER, H2, INP, SIGNAL, S2, FUNS, SEQ, SEQ$01, FUN.

Имя H2 взято из выходной ссылки, S2 - из структуры данных, а SEQ$01 - новое имя.

2) FILTER, H2, TF, TF, T2, COEFF

Имя H2 взято из выходной ссылки, а T2 - из структуры данных.

3. Необходимо найти отношение для вычисления
FILTER, H2, INP, SIGNAL, S2, FUNS, SRQ, SEQ$01, FUN.

Выбираем отношение SEQM, так как ссылка из описания отношения FILT не входит в ссылку из данных.

4. Формируем входные ссылки

I) FILTER, H2, INP, SIGNAL, S2, FUNS, SIN, SIN$01, AMR

2) FILTER, H2, INP, SIGNAL, S2, FUNS, SIN, SIN$01, EREQ.

Для вычисления этих атрибутов отношений нет и в структуре данных их также нет, так что возвращаемся к FILTER, H2, INP, SIGNAL, S2, FUNS, SEQ, SEQ$01, FUN.

5. Выбираем отношение FILT и формируем новую выходную ссылку FILTER, H1, OUT, SIGNAL, S2, FUNS,SEQ,SEQ$01,FUN. Подчеркнутая часть изменена, имя H1 взято из структуры данных.

6. Формируем входные ссылки

I) FILTER, H1, INP, SIGNAL, S1, FUNS, SEQ, SEQ$02, FUN

2) FILTER, H1, TF, TF, T1, COEFF

7. Для нахождения FILTER, H1, INP, SIGNAL, S1, FUNS, SEQ, SEQ$02, FUN

выбираем отношение SEQM.

8. Формируются входные ссылки

I) FILTER, H1, INP, SIGNAL, S1, FUNS, SIN, AMPL

2) FILTER, H1, INP, SIGNAL, S1, FUNS, SIN, FREQ

На этом планирование заканчивается.

6. Заключение

Рассмотренные в настоящей статье алгоритмы реализованы в планировщике системы SPADE и прошли этап опытной эксплуатации. Несмотря на то, что в задачах анализа электронных схем могут быть спланированы лишь части вычислительного процесса, не включающие итерации, автоматическое планирование дает все же достаточно большой эффект.

Согласование данных в задаче с описанием модели оказывается практически самой сложной проблемой планирования и

решение ее с помощью иерархических структур представляется наиболее простым и естественным.

Литература

1. Тыугу Э.Х. Решение задач на вычислительных моделях. - ЭВМ и МФ, т. 10, 1970, № 3, с. 716-733.

2. Тыугу Э.Х. Решатель вычислительных задач. -ЖВМ и МФ, т. 11, 1971, № 4, с. 992-1002.

3. Кукк В. Принципы функционирования системы. - Тр. Таллинск. политехн. ин-та, 1981, № 535.

4. Венделин Е. Структуры данных и входной язык. - Тр. Таллинск. политехн. ин-та, 1982, № 535.

5. Кукк В., Венделин Е. Управление вычислительным процессом. - Тр. Таллинск. политехн. ин-та, 1982, № 535.

E. Vendelin

IDENTIFICATION OF DATA OBJECTS IN COMPUTATION PROCESS PLANNING

Summary

The paper deals with problems of computation process planning using computer models. Application of circuits and control systems is discussed.

A model consists of two parts: data objects and relations. In our case there are two levels of models: the area model and the task model. Area model consists of object types and relations between these types. Task model is a set of objects with types described in area model and concrete values. The main problem in computation process planning is to match relations from the area model to the objects of the task.

№ 550

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED

ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

УДК 62-507

П. Лейс, К. Салум

ИТЕРАТИВНЫЙ МЕТОД ДЕКОМПОЗИЦИИ КОНЕЧНЫХ АВТОМАТОВ

I. Введение

В последнее время наблюдается увеличение числа работ, посвященных разработке различных декомпозиционных методов синтеза конечных автоматов. Это вызвано, в первую очередь, тем, что канонические методы синтеза конечных автоматов оказываются неудовлетворительными в условиях резкого роста сложности синтезируемых автоматов и перехода на новый элементный базис в виде больших интегральных схем.

Настоящая работа является продолжением работ [I, 2]по декомпозиции конечных автоматов. В [I] был предложен метод декомпозиции конечного автомата по ортогональному множеству разбиений на его множестве состояний, а в [2] на этой основе разработаны эффективные алгоритмы декомпозиции автоматов большой сложности (микропрограммных автоматов). Структура сети (соединения между компонентными автоматами) определяется свойствами ортогонального множества разбиений на множестве состояний декомпозируемого автомата.

Известны алгоритмы для нахождения ортогонального множества разбиений, по которым можно построить параллельную, последовательную или последовательно-параллельную сеть. Однако не существует метода для построения сети с произвольной заданной структурой. Кроме того, даже в случаях, когда в принципе существуют методы нахождения ортогонального множества разбиений с нужными свойствами, их практическое применение оказывается невозможным из-за огромных вычислительных трудностей. Для преодоления этих трудностей в данной работе предлагается итеративный метод декомпозиции конечных автоматов.

2. Основные понятия

Введем символику и основные понятия, используемые в настоящей работе. Поскольку вопросы, связанные с выработкой выходных сигналов, здесь не рассматриваются, то определим автомат как тройку $A=(I,S,\delta)$, где I - конечный входной алфавит; S - конечное множество состояний; $\delta: D(\delta)\to S$ - функция переходов, $D(\delta)\subseteq I\times S$. Сеть автоматов $N=(B,I,\Phi)$ определяется следующим образом:

1^0 $B=\{A_i \mid i\in J=\{1,\dots,n\}\}$ - множество компонентных автоматов (базис сети),

$$A_i=(S_i,I_i,\delta_i),$$

здесь $I_i=I_i'\times I_i''$ где $I_i'\subseteq I$ - внешний входной алфавит, I_i'' - внутренний входной алфавит.

2^0 $\Phi=\{\varphi_i \mid i\in J\}$ - множество функций соединения (структура сети),

$$\varphi_i: \underset{j\in J_i}{\times} A_j \to I_i'', \quad J_i\subseteq J.$$

Каждой сети N можно сопоставить автомат A_N, функционально эквивалентный N [1]. Сеть N называется декомпозицией автомата A, если и только если у автомата A_N имеется подавтомат A_N', изоморфный автомату A.

Введем операции с покрытиями и разбиениями.

Покрытием φ на множестве S называется подмножество множества 2^S, удовлетворяющее условиям:

1^0 $\bigcup\limits_{B\in\varphi} B=S$

2^0 $(\forall B\in\varphi)(\forall B'\in\varphi)[B\subseteq B' \Rightarrow B=B']$.

Если дополнительно $B\cap B'=\emptyset$ для всех $B\in\varphi, B'\in\varphi$, то φ называется разбиением на S. Введем обозначение:

$$a \underset{\varphi}{\equiv} b \Leftrightarrow (\exists B\in\varphi)[a\in B \,\&\, b\in B].$$

Произведением покрытий (разбиений) π и τ называется покрытие (разбиение)

$$\varphi=\pi\cdot\tau=\max(\{B'\cap B \mid B'\in\pi \,\&\, B\in\tau\}),$$

где оператор $\max$ удаляет из φ блоки, содержащиеся в других блоках.

Сумма покрытий π и τ определяется следующим образом:

$$\pi + \tau = \max(\pi \cup \tau).$$

На множестве всех покрытий (разбиений) вводится отношение частичного порядка:

$$\tau \leqslant \pi \Leftrightarrow \tau \cdot \pi = \tau.$$

3. Структурная матрица сети автоматов

В целях формального исследования структуры сети, что является основной задачей данной работы, введем аппарат структурных матриц сети автоматов [I].

Структурной матрицей сети N называется матрица $\|m_{ij}\|$, $i \in J$, $j \in J$, определенная следующим образом:

$$m_{ij} = \begin{cases} 1, & \text{если } j \in J_i; \\ 0, & \text{если } j \notin J_i. \end{cases}$$

Содержательно $m_{ij}=1$, если поведение A_i существенно зависит от A_j, т.е. в сети A_i соединен с A_j.

Пусть $P=\{\pi_i | i \in J\}$ – множество ортогональных разбиений, т.е. $\prod\limits_{i \in J} \pi_i = 0$. При декомпозиции автомата A по P в сети каждому разбиению $\pi_i \in P, i \in J$ соответствует компонентный автомат A_i.

Для определения структуры сети используется аппарат алгебры пар [3], основным понятием которой является понятие пары разбиений.

Разбиения τ и π на множестве состояний S образуют пару разбиений (τ, π), если и только если

$$s' \underset{\tau}{\equiv} s''' \Rightarrow \delta(s', x) \underset{\pi}{\equiv} \delta(s'', x)$$

для всех $x \in I$; $s', s'' \in S$.

В [3] показано, что для заданного разбиения π существует максимальное разбиение $M(\pi)$, образующее пару разбиений с π. В случае частично определенных автоматов аппарат пар разбиений является уже недостаточным и необходимо использовать аппарат смешанных пар [4].

Покрытие φ и разбиение π на множестве состояний образуют смешанную пару (φ, π), если и только если

$$s' \underset{\varphi}{\equiv} s'' \Rightarrow \delta(s', x) \underset{\pi}{\equiv} \delta(s'', x)$$

для всех (s', x), $(s'', x) \in D(\delta)$.

Существует максимальное покрытие $M_c(\pi)$, образующее с заданным разбиением π смешанную пару. Отметим, что для полностью определенных автоматов справедливо $M_c(\pi) = M(\pi)$, [4].

Теперь для определения структуры сети, построенной по множеству P, можно пользоваться следующим отношением [4]:

$$M_c(\pi_i) \geq \prod_{j \in J_i} \pi_j, \; i \in J, \; J_i \subseteq J. \qquad (I)$$

При этом для нахождения сети с минимальным числом связей между компонентными автоматами необходимо найти множества $J_i \subseteq J$, $i \in J$ с минимальным числом элементов. Другими словами, заданному автомату A и множеству P соответствует множество структурных матриц $\{\|m_{ij}\|\}$, из которых обычно необходимо найти матрицу с минимальным числом единиц. Решение этой задачи обсуждалось в [I].

Приведем два примера, иллюстрирующих использование структурной матрицы для определения свойств разбиений из P, соответствующих заданной структуре сети.

Пример I. Параллельная сеть

Нетрудно видеть, что в этом случае $m_{ii}=1; m_{ij}=0$, если $i \neq j$ [5], т.е. $J_i = \{i\}$ для всех $i \in J$.

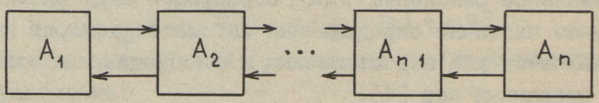
Теперь непосредственно из (I) получаем, что

$$M_c(\pi_i) \geq \pi_i, \; i \in J, \qquad (2)$$

т.е. разбиения $\pi_i, i \in J$ являются SP-разбиениями [3].

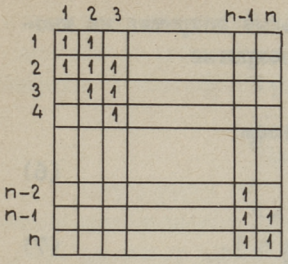
Пример 2. Линейная сеть

Сеть автоматов называется линейной, если компонентные автоматы $A_i, i \in J$ можно упорядочить таким образом, что между собой связаны только автоматы A_i и A_{i+1}, $1 \leq i \leq n-1$. Линейная четь представлена на фиг. I.



Фиг. I.

Вполне очевидно, что структурная матрица линейной сети заполнена следующим образом:



В силу этого из (I) получим:

$$\left.\begin{aligned} M_c(\pi_1) &\geq \pi_1 \cdot \pi_2 \\ M_c(\pi_2) &\geq \pi_1 \cdot \pi_2 \cdot \pi_3 \\ &\cdots \\ M_c(\pi_i) &\geq \pi_{i-1} \cdot \pi_2 \cdot \pi_{i+1} \\ &\cdots \\ M_c(\pi_n) &\geq \pi_{n-1} \cdot \pi_n \end{aligned}\right\} \qquad (3)$$

Фиг. 2.

Рассмотренные примеры подтверждают, что получение отношений типа (2), (3) по (I) не вызывает трудностей и для сети с любой заданной структурой. Однако эти отношения не зададут алгоритма для вычисления соответствующего множества разбиений P. Одно решение последней задачи рассматривается в следующем пункте.

4. Решение декомпозиционных неравенств

Будем называть декомпозиционными неравенствами (ДР) неравенства следующего вида:

$$\tau_j \geq \prod_{k \in K} \pi_k \prod_{l \in L} \rho_l, \quad j \in J, \qquad (4)$$

где τ_j – известное покрытие или разбиение;

$\pi_k, k \in K$ – разбиения;

$\rho_l, l \in L$ – покрытия.

Некоторые разбиения π_k и покрытия ρ_l известны, остальные требуется определять так, что выполняются неравенства (4). Здесь J, K, L – множества индексов.

В качестве примера рассмотрим одно простое ДР:

$$\tau \geq \pi_1 \cdot \pi_2, \qquad (5)$$

где τ и π_1 известны, необходимо найти π_2. Формально мы можем написать: $\pi_2 \leq \frac{\tau}{\pi_1}$.

Отсюда ясно, что для решения ДР необходимо ввести операцию деления для разбиений и покрытий. Соответствующий математический аппарат был предложен в литературе [6], откуда приводим нужные нам понятия и результаты.

Пусть далее α, β и γ – произвольные покрытия на множестве S. Частным α и β называется покрытие

$$\frac{\alpha}{\beta} = \sum\{\gamma \mid \gamma \cdot \beta = \alpha\},$$

которое можно вычислить следующим образом:

$$\frac{\alpha}{\beta} = S/R(\alpha,\beta), \tag{6}$$

где

$$R(\alpha,\beta) = \{(s,s') \mid s \underset{\alpha}{\equiv} s' \vee s \underset{\beta}{\not\equiv} s'\}. \tag{7}$$

Здесь $S/R(\alpha,\beta)$ – фактор-множество отношения толерантности (рефлексивного и симметричного отношения) $R(\alpha,\beta)$.

Справедливы следующие соотношения:

$$\frac{\alpha}{\beta} \cdot \beta = \alpha \cdot \beta \tag{8}$$

$$\frac{\alpha \cdot \beta}{\beta} = \frac{\alpha}{\beta} \tag{9}$$

$$\gamma \cdot \beta \leq \alpha \Leftrightarrow \gamma \leq \frac{\alpha}{\beta} \tag{10}$$

$$\gamma \geq \beta \Leftrightarrow \frac{\alpha}{\gamma} \leq \frac{\alpha}{\beta}. \tag{11}$$

Из (10) получим, что решением ДР (5) является $\pi_2 \leq \frac{\tau}{\pi_1}$, где частное вычисляется по (6). Отметим, что при этом π_2 может оказаться покрытием. Если необходимо получить решением ДР (5) разбиение, что имеет место при решении задачи декомпозиции средствами из [1, 2, 4], то нужно найти некоторое разбиение графа отношения $R(\tau,\pi_1)$ на минимальное число полных подграфов. Отметим, во-первых, что эта задача NP-полна, и во-вторых, таких разбиений может быть несколько. Поэтому можно заключить, что точное решение ДР связано с большими вычислительными трудностями. Однако при решении задач декомпозиции нет необходимости найти точное решение ДР, а лишь достаточно близкое к точному решение. Но нахождение такого решения может иметь существенно меньшую временную сложность. В данной работе проблема приближенного решения ДР и создание соответствующих алгоритмов и программ не рассматривается.

Продолжим рассмотрение примера 2. Если π_1 известно, то из (3), используя (10), получим:

$$\left.\begin{aligned}
&\pi_2 \leq \frac{M_C(\pi_1)}{\pi_1} \\
&\pi_3 \leq \frac{M_C(\pi_2)}{\pi_1 \cdot \pi_2} \\
&\dots \\
&\pi_{i+1} \leq \frac{M_C(\pi_i)}{\pi_{i-1} \cdot \pi_i} \\
&\dots \\
&\pi_n \leq \frac{M_C(\pi_{n-1})}{\pi_{n-1}}
\end{aligned}\right\} \qquad (I2)$$

Систему (I2) можно интерпретировать как итеративную схему нахождения линейной декомпозиции с заданным исходным разбиением π_1. Отметим, что работа [7] посвящена проблеме нахождения линейной декомпозиции конечного автомата с помощью итеративной схемы. Но несмотря на доказательство множества теорем и лемм, получить очевидных результатов (I2) в этой работе не удалось. Благодаря использованию более совершенного математического аппарата структурных матриц и деления покрытий, мы эти результаты получили почти автоматически.

Ясно, что сложным структурам сети могут соответствовать ДР, не допускающие решения с помощью итеративной схемы. Например, ДР может содержать несколько переменных, значения которых необходимо определить на одном шаге итерации и т.д. Для решения таких задач предлагаем следующий метод.

Заданный автомат A декомпозируется на два автомата A_1 и A_2. Далее, в соответствии с выбранной структурой, декомпозируется автомат(ы) A_1 и/или A_2 и т.д. Такой метод позволяет на каждом шаге итерации решать ДР небольшой сложности. Недостатком метода является то, что, выбирая на каждом шаге оптимальное решение, мы не можем гарантировать оптимальность решения в целом.

Предложенный метод применим и в случае, когда из-за большой размерности декомпозируемого автомата известные методы нахождения ортогонального множества разбиений становятся непригодными.

5. Заключение

На основе структурных матриц сетей автоматов и деления покрытий предложен итеративный метод декомпозиции конечных автоматов, позволяющих построить сети с заданной структу-

рой. Из-за итеративного характера метод применим также для автоматов очень большой размерности. Необходимы дальнейшие исследования для создания приближенных алгоритмов и программ решения декомпозиционных неравенств.

Литература

I. Кеэваллик А.Э. Теорема декомпозиции конечных автоматов. - Автоматика и вычислительная техника, 1974, № 1, с. 17-24.

2. Jakobson G., Keevallik A., Leis P. Some aspects of the construction of microprogram automata networks. - In: IFAC-symposium "Discrete Systems". Dresden, 1977, p. 120-128.

3. Hartmanis J., Stearns R.E. Algebraic Structure Theory of Sequential Machines. - Englewood Cliffs, N.-Y., Prentice-Hall Inc., 1966. 211 p.

4. Лейс П. Определение структуры сети при декомпозиции неполностью определенных микропрограммных автоматов.- Тр. Таллинск. политехн. ин-та, 1977, № 432, с. 57-68.

5. Кеэваллик А.Э. О классификации задач декомпозиции конечных автоматов. - Тр. Таллинск. политехн. ин-та, 1973, № 350, с. 61-67.

6. Zech K.-A. Feedback-orientierte Kodierung endlicher Automaten. - Elektronische Information - Verarbeitung und Kybernetik. 1973, 9, N 3, S. 635-650.

7. Poe A.T. Linear N-decompositions and tri-diagonal matrices. - Information and Control, 1977, vol. 34, No. 3, p. 210-221.

P. Leis, K. Salum

THE METHOD OF ITERATIVE DECOMPOSITION OF FINITE AUTOMATA

Summary

In this paper the decomposition method of finite automata which allows to construct the automata network with any given structure is developed. The method is based on the concept of structure matrix of automata network. The iterative character of the method enables to use it for the synthesis of automata with very large dimensions.

№ 550

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED

ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

УДК 681.142.71

А.Э. Кеэваллик, П.А. Китсник,
А.В. Судницын

МЕТОД ПРОГРАММНОЙ РЕАЛИЗАЦИИ МИКРОПРОГРАММНОГО АВТОМАТА

В последнее время особую актуальность получило проектирование дискретных управляющих автоматов на основе микропроцессорных БИС. Микропроцессорную систему можно рассматривать как универсальный программируемый модуль, позволяющий реализовать различные алгоритмы управления. Существует два принципиально различных направления программной реализации управляющих автоматов [1, 2]. В одном из них в каждом конкретном случае осуществляется переход от алгоритма функционирования автомата к реализующей его программе (компиляционный подход). В другом случае автомат вводится в память как данные в виде некоторого регулярного списка, а его управляющее воздействие находится при помощи одинаковой для всех автоматов программы (интерпретационный подход). Методы интерпретации позволяют довести затраты на программирование до минимума, а также в определенных случаях повысить быстродействие программной реализации. Их применение целесообразно и тогда, когда возникает необходимость частой перенастройки устройства управления.

В настоящей работе предполагается интерпретационный метод программной реализации микропрограммных автоматов (МПА), который распространен также для реализации сетей из МПА, полученных в результате декомпозиции заданного МПА.

Формально под МПА [3] можно понимать пятерку $A=(S, X, Y, \delta, \lambda)$,

где $S=\{s_1, s_2, \ldots, s_M\}$ - множество состояний,
$X=\{x_1, x_2, \ldots, x_L\}$ - множество входных двоичных переменных,
$Y=\{y_1, y_2, \ldots, y_N\}$ - множество выходных двоичных переменных,

$\delta: S \times \{0,1\}^L \to S$ - функция переходов,
$\lambda: S \times \{0,1\}^N$ - функция выходов (модель Мура).

Показано [2], что к самым эффективным по быстродействию программным реализациям автоматной модели приводит метод, суть которого заключается в следующем. Автомат представляется в памяти массивом пар слов (из слова следующего состояния и слова значений выходных переменных). Считываемое извне слово входных переменных суммируется со словом текущего состояния. По полученному адресу считывается пара слов из следующего состояния и выходного слова, которое выдается на порт вывода. При непосредственном применении метода для реализации МПА затраты на память оцениваются как $M \cdot 2^L (N + \lceil \log_2 M \rceil)$ бит, а быстродействие как

$$C \cdot \left\lceil \frac{L + \lceil \log_2 M \rceil}{q} \right\rceil,$$

где q - длина обрабатываемых командой операндов и
C - константа, определяемая временем вычисления выходного сигнала данным микропроцессором.

Очевидно, что уже для сравнительно несложных МПА (несколько десятков двоичных входов и выходов и состояний) этот метод из-за чрезмерных затрат памяти практически непригоден. Поэтому предлагаем модификацию изложенного метода суммирования, которая опирается на учет некоторых количественных соотношений между параметрами МПА. Поскольку в реальных МПА количество существенных на переходе δ_i в δ_j переменных составляет лишь незначительную долю от количества L всех входных переменных [3], то можно путем организации соответствующего логического интерфейса существенно сократить затраты на память программной реализации.

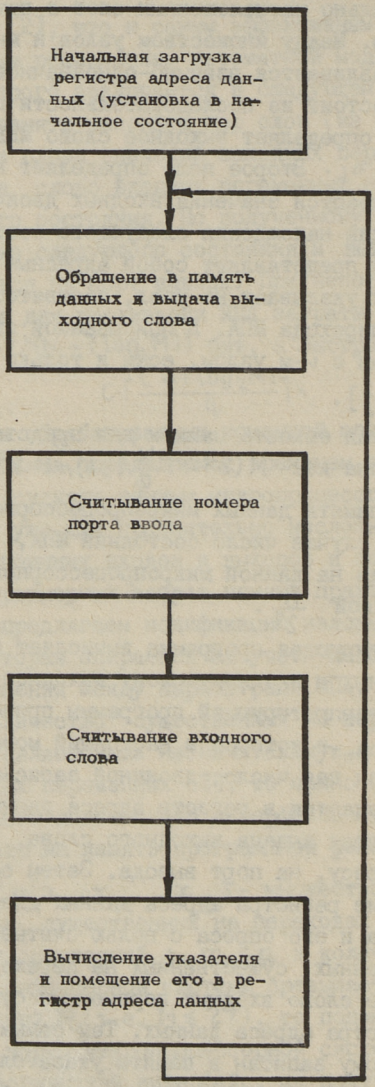
Пусть $X_m = \{x_{m_1}, x_{m_2}, \ldots, x_{m_K}\}$ - множество входных двоичных переменных, существенных на переходах из состояния δ_m. Если для всех состояний δ_m $m_K \le q$ и количество адресуемых портов микропроцессорной системы превышает количество внутренних состояний M (т.е. $M \le 2^q$), то подаем множества X_m на отдельные порты микропроцессорной системы. Образуя таким образом логический интерфейс, мы получаем возможность считывать необходимые входные сигналы за один такт, что, в свою очередь, влечет за собой и сокращение объема памяти программы.

Информационная таблица, задающая функции переходов и выходов МПА, представляется совокупностью узлов, состоящих из последовательно расположенных слов в памяти микропроцессорной системы. Между множеством узлов и множеством состояний МПА устанавливается взаимно-однозначное соответствие. Каждый узел состоит из последовательности полей. Первое поле i-го узла определяет выходное слово МПА, соответствующее состоянию a_i. Второе поле определяет номер порта, с которого считываются значения входных двоичных переменных, существенных для нахождения следующего состояния МПА. Последующие поля представляют собой адресные указатели, каждый из которых указывает на узел, соответствующий некоторому состоянию перехода МПА. В описываемой структуре данных j-й узел связан с i-м узлом, если и только если $\exists \Delta \in \{0,1\}^L$ $[\delta(a_i, \Delta) = a_j]$.

Необходимая емкость памяти для представления МПА может быть оценена как $M(2^q + \lceil \frac{N}{q} \rceil + 1) \sim M \cdot 2^q$ слов памяти. Если емкость памяти данных микропроцессорной системы равна R, то в общем случае число состояний МПА, который может быть реализован на данной микропроцессорной системе, ограничено величиной $\frac{R}{2^q}$.

Интерпретирующая программа вычисляет по описанной структуре данных и входному слову выходное слово автомата. Блок-схема интерпретирующей программы приведена на фиг. I. Код состояния a_m, в котором в настоящий момент находится МПА, рассматривается как число, двоичной записью которого он является, и помещается в регистр адреса данных. На первом шаге осуществляется выдача выходного слова, расположенного по указанному адресу, на порт вывода. Затем осуществляется инкрементирование регистра адреса данных для нахождения номера порта ввода и его опроса с целью считывания значений входных переменных, существенных на переходе МПА из a_m. Считываемое извне слово входных переменных суммируется с содержимым регистра адреса данных. Тем самым образуется адрес, по которому записан в памяти указатель, соответствующий следующему состоянию автомата, который, в свою очередь, помещается в регистр адреса данных.

Необходимый объем памяти можно уменьшить, если в интерпретирующую программу включить операцию маскирования, ко-


Начальная загрузка регистра адреса данных (установка в начальное состояние)
Обращение в память данных и выдача выходного слова
Считывание номера порта ввода
Считывание входного слова
Вычисление указателя и помещение его в регистр адреса данных

Фиг. 1.

торая следует сразу за опросом порта ввода. Под маскированием вектора $\vec{v}$ по маске $\vec{b}$ понимается операция по компонентной конъюнкции логических векторов $\vec{v}$ и $\vec{b}$. В качестве маски в m-том узле структуры данных используется булев вектор, единичное значение у которого принимают компоненты, соответствующие входным переменным, существенным для перехода из m-того состояния.

Необходимое число портов для реализации зависит от организации логического интерфейса, т.е. способа объединения (упаковки) входных битовых переменных в байты (в случае использования 8-разрядного микропроцессора). Со стороны практики выдвигается задача минимизации числа используемых портов, которую можно рассматривать как задачу разбиения множества $\{\tilde{X}_m / m = 1,\ldots,M\}$ на минимальное число классов так, чтобы общее число входных переменных каждого класса не превышало q_j. Один из методов решения такой задачи можно найти в работе [4].

До сих пор мы полагали, что для всех состояний $m_k \leq q_j$. Если это не так, то можно декомпозировать реализуемый МПА на сеть МПА [5], для каждого компонентного автомата A_i, которой это условие было бы выполнено. Полученная в результате декомпозиции сеть МПА может быть реализована мультимикропроцессорной системой с общей магистральной шиной, в которой каждый компонентный автомат и выходная функция сети реализованы на отдельном микропроцессоре. Поскольку каждый компонентный автомат сети имеет по два входных алфавита (внутренний и внешний), то необходима предварительная обработка входной информации. В случае реализации компонентного автомата A_i интерпретационным методом, в начале его описания расположим вектор указателей из M слов (по числу состояний заданного МПА). Принимаемое внутреннее входное слово (результирующее состояние сети) рассматриваем как адрес указателя, определяющего переход по внутреннему входу компонентного автомата. Указатель помещаем в регистр адреса данных и складываем с принимаемым словом внешних входных переменных, существенных для вычисления следующего состояния. По полученному адресу расположим слово, идентифицирующее следующее состояние компонентного автомата, которое передается на межпроцессорную шину данных.

Отметим, что в данном случае декомпозиционный метод синтеза не приводит к повышению быстродействия программной реализации. Однако при компиляционном подходе соответствующие декомпозиционные методы позволяют распараллеливать вычисления и тем самым повысить быстродействие программной реализации [5].

Литература

I. Кузнецов О.П. О программной реализации логических функций в автоматов. - Автоматика и телемеханика, 1977, № 7, с. 163-174, № 9, с. 137-148.

2. Пупырев Е.И. Интерпретирующие программы реализации булевых функций и автоматов. - Автоматика и телемеханика, 1982, № 1, с. 132-140.

3. Баранов С.И. Синтез микропрограммных автоматов (граф-схемы и автоматы). Энергия, 1979. 232 с.

4. Закревский А.Д. Логический синтез каскадных схем. М., Наука, 1981. 416 с.

5. Кеэваллик А.Э., Судницын А.В. Использование методов декомпозиции конечных автоматов для построения сетей микропроцессоров. - В кн.: Прикладные аспекты теории автоматов. Тр. У Международного семинара. Болгария, Варна, 1979, т. 1, с. 54-65.

A. Keevallik, P. Kitsnik,
A. Sudnitsyn

THE PROGRAM REALIZATION OF MICROPROGRAM AUTOMATA

Summary

The paper deals with the formal method for binary input - binary output control automata implementation using microprocessor-based systems. A general device - independent interpretative program has been proposed which enables to identify the next state and the output word of automata. It has been shown that the operating speed of the microprocessor system can be increased using special decomposition techniques.

№ 550

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED
ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

УДК 681.32

Р. Убар

ОБОБЩЕННАЯ МОДЕЛЬ АЛЬТЕРНАТИВНЫХ ГРАФОВ ДЛЯ СИНТЕЗА ТЕСТОВ ЦИФРОВЫХ СИСТЕМ

I. Введение

Трудности использования известных структурных методов синтеза тестов для микропроцессорных БИС обусловлены большой размерностью математических моделей, основанных на вентильном представлении контролируемых схем.

В данной статье предлагается обобщенная модель альтернативных графов (АГ) с целью автоматизации процесса синтеза тест-программ для нерегулярных БИС и микропроцессорных систем.

В работах [I-4] рассмотрен аппарат альтернативных графов, обеспечивающий более сжатое описание цифровых схем, чем традиционный язык булевых выражений, а также предложены методы синтеза тестов, базирующиеся на аппарате АГ. Ниже показываются пути дальнейшего развития модели АГ с целью автоматизации процесса синтеза тест-программ для сложных цифровых объектов, в частности, микропроцессорных БИС и систем.

В основе предлагаемой обобщенной модели АГ лежит возможность работы не только на множестве двоичных сигналов, а также на множестве более "крупных" переменных, в частности, групп сигналов, рассматриваемых как единые целые и разнесенные как в пространстве, так и во времени. Такими переменными могут служить, например, двоичные векторы (операнды, данные), целочисленные переменные (коды команд, коды состояний микропрограммных автоматов, адреса и т.д.), последовательности сигналов (пачки импульсов и т.д.).

2. Переменные модели

Рассмотрим объект диагностирования (ОД) в виде множества функций

$$z_k = f_k(Z_k), \quad z_k \in Z, \qquad (I)$$

где Z конечное множество переменных, используемых для описания ОД, а $Z_k \subseteq Z$ некоторое подмножество Z. Функции (I) могут быть заданы произвольным принятым образом (например, в виде булевых выражений, таблиц истинности, таблиц переходов, временных диаграмм, микропрограмм, выражений на различных языках и т.д.). В дальнейшем будем предполагать, что функционирование ОД происходит в дискретном времени, где значение z_k относится к некоторому текущему моменту t, а значения $z \in Z_k$ относятся к предыдущему моменту $t-I$. Промежутки между t и $t-I$ могут покрывать такты, циклы микрокоманд, циклы команд и т.д. в зависимости от уровня описания ОД.

Множество переменных Z введено для описания входов, выходов и внутренних состояний ОД. В общем случае переменные $z \in Z$ могут иметь формат $z(n_2 : n_1)$, где n_1 и n_2 – соответственно, номера младшего и старшего двоичных разрядов слова, представляемых переменной, $n_1 \leq n_2$. В частном случае простых булевых переменных имеем $n_1 = n_2 = I$. При выделении в слове z_k одноразрядных полей используем обозначение $z_{k,i}$, где i -номер рассматриваемого разряда, $n_1 \leq i \leq n_2$. Переменными $z \in Z$ могут быть представлены в ОД магистрали, регистры, предназначенные для хранения данных, команд, адресов, шины для передачи синхросигналов, управляющих воздействий и т.д.

Обозначим через $S(z)$ множество значений переменной z. Тогда множество

$$S(f_k) = \bigcup_{z \in Z_k} S(z) \qquad (2)$$

образует область определения функции $z_k = f_k(Z_k)$.

Все переменные $z \in Z$ делятся на два класса: дискретные и аналоговые. Дискретные переменные делятся в модели на одноразрядные и многоразрядные, а последние, в свою очередь, на двоичные векторы и целочисленные переменные.

Одноразрядные переменные $z_к$ используются для представления двоичных потенциальных сигналов (в частности, выходных сигналов управляющих автоматов (УА), осведомительных сигналов операционных автоматов (ОА), сигналов прерывания и т.д.). Областью их определения является $S(z_к) = \{0,I\}$.

Под двоичными векторами $z_к(n_2 : n_1)$ понимаем многоразрядные переменные, отдельные разряды которых $z_{к,i}$, $n_2 \leqslant i \leqslant n_1$, в функциональном смысле независимы друг от друга. Двоичными векторами представляются, например, регистры, предназначенные для хранения данных и операндов. Векторные переменные введены для создания возможности проведения вычислений параллельно для всех или некоторой группы разрядов рассматриваемого компонента ОД. Отдельные компоненты $z_{к,i}$ двоичных векторов принимают значения из множества $S(z_{к,i}) = \{0,I\}$.

Целочисленными переменными можно обозначить в модели функционально законченные подсхемы, состоящие из регистра и дешифратора и предназначенные для задания команд, микрокоманд, состояний микропрограммных автоматов, адресов и т.д. Каждому выходу дешифратора соответствует здесь определенный двоичный код, который может быть интерпретирован как некоторое десятичное число. Целочисленные переменные целесообразно ввести с целью сокращения числа переборов при синтезе тестов: вместо генерирования определенного кода бит за битом проще генерировать код как одно целое. Целочисленные переменные $z_к(n_2 : n_1)$ принимают значения из множества целых чисел $S(z_к) = \{0,I,2,\ldots n\}$, где $n = 2^{n_2 - n_1} - I$.

Под аналоговыми переменными понимаем в модели импульсные сигналы и импульсные последовательности. Импульсные переменные принимают значения из множества $\{0,I\}$, где 0 обозначает отсутствие данного вида импульса и I - его присутствие. Импульсные последовательности задаются через целочисленные переменные, значения которых определяют число импульсов в последовательности. Переменные последовательностей введены с целью повышения производительности при синтезе тестов, в частности, для создания возможности рассматривания пачки импульсов как простого компонента одного входного набора. Так, например, при синтезе последовательности из n импульсов, вместо синтеза n двоичных наборов, до-

статочно синтезировать всего лишь один набор, где переменная последовательности принимает значение n.

3. Обобщенные альтернативные графы

Обобщенные альтернативные графы введены для описания функций $z_k = f_k(Z_k)$ структурных компонентов рассматриваемого ОД. Такими компонентами могут быть, например, произвольные функционально законченные части объекта (если объект рассматривается на схемном уровне), программно- и микропрограммно-доступные элементы (если объект рассматривается, соответственно, на программном или микропрограммном уровнях) и пр.

Рассмотрим АГ, описывающий функцию $z_k = f_k(Z_k)$ в виде пары $G_k = \{M_k, \Gamma_k\}$, где M_k - некоторое конечное множество вершин, а Γ_k отображает M_k в M_k. Вершины $m \in M_k$ взвешены переменными $z(m) \in Z_k$. Каждому значению $z(m) \in S$ $[z(m)]$ соответствует определенный выход из вершины m, т.е. вход в некоторую другую вершину $m' = \Gamma_k(m, z(m)) \in \Gamma_k(m)$, являющуюся последователем m. Здесь $\Gamma_k(m)$ - множество всех последователей m. В частном случае, если $\Gamma_k(m, z(m)) = \phi$, происходит выход из графа. Число выходов из каждой вершины равно числу различных значений соответствующей весовой переменной из ее области определения. Так, например, дешифратор, имеющий n выходов, описывается в модели вершиной, имеющей также n выходов.

Рассмотрим некоторый путь l на графе G_k, из начальной вершины графа в вершину $m \in M_k$, проходящий через некоторое подмножество вершин $M'_k \subseteq M$. Легко заметить, что каждому пути можно сопоставить определенную совокупность значений переменных $z(m) \in Z_k$, где $m \in M'_k$, таких, чтобы значения $z(m)$ определили движение по пути l, согласно отображению $\Gamma_k(m, z(m))$.

Назовем далее процесс синтеза совокупности значений переменных, порождающих некоторый путь l на графе G_k, процессом активизации пути.

Приписав теперь каждому выходу графа G_k некоторое значение $z^*_k \in S(z_k)$, создана возможность отображения в АГ функции $z_k = f_k(Z_k)$.

Определение. АГ G_K описывает функцию $z_K = f_K(Z_K)$, если любой набор Z_K^* значений переменных $z \in Z_K$, приводящий к значению $z_K^* \in S(z_K)$, порождает путь из начальной вершины графа к его выходу, к которому также приписано значение z_K^*.

Учитывая вышесказанное, задачи, связанные с синтезом и анализом тестов, приводимы к задачам выбора и прохождения путей на АГ. Так, например, задача моделирования заданного входного набора Z_K^*, с целью определения значения z_K сводится на АГ к прохождению некоторого пути, согласно значениям переменных $z(m) \in Z_K$, с целью определения выхода из графа и приписанного этому значения z_K^*. Обратная же задача синтеза некоторого установочного набора Z_K^*, устанавливающего определенное значение z_K^* для переменной z_K, сводится к поиску некоторого пути l на графе, приводящему к выходу графа, имеющему значение z_K^*. Построенный путь порождает искомый набор Z_K^*.

Заметим, что пути на АГ не соответствуют сигнальным путям между структурными компонентами ОД в традиционном смысле. Пути АГ задают отношения порядка различных сигналов объекта, позволяющие организовать целенаправленный перебор различных функциональных режимов. Структурные связи между компонентами объекта задаются на модели АГ через множество переменных Z.

С целью разработки эффективных алгоритмов ситнтеза и анализа тестов, отличающихся направленным поиском решений и уменьшенным перебором безрезультативных вариантов, введем так называемый класс двоичных АГ. Это графы, висячие вершины которых имеют только два выхода, причем одному из них (условно, правому) приписано значение I, а другому (условно, направленному вниз) - значение 0.

Если речь идет о представлении функций $z_K = f_K(Z_K)$, где $S(z_K) = \{0,I\}$, то графы можно строить так, чтобы значения переменных $z(m)$ для висячих вершин определяли значение z_K. Если z_K - векторная переменная, то задачу выбора путей на графе можно понимать как задачу синтеза установочных наборов для некоторого обобщенного разряда $z_{K,i}$. Полученные наборы можно затем "разнести в пространстве" и для других разрядов $n_2 \leq i \leq n_1$. С другой стороны, относительно

других разрядов могут быть решены и другие установочные задачи, дающие или противоположное значение $z_{к,i}$, или то же самое значение, но при измененных ограничениях. В первом случае осуществляется на графе одновременное решение многих идентичных задач, во втором случае - последовательное решение различных задач с последующим сжатием результатов в один набор.

При моделировании заданных наборов на графе $G_к$, вершины m которого взвешены векторными переменными $z(m)$, требуется введение двух масок M_1 и M_0 для выделения разрядов, для которых движение разветвляется в разных направлениях

$$M_1 = M(m)\,z(m), \quad M_0 = M(m)\,\overline{z(m)}, \qquad (3)$$

где $M(m)$ – текущая маска вершины m для выделения разрядов, относительно которых вершина m достигнута. Здесь M_1 и M_0 служат текущими масками, соответственно, для вершин $m^1 = \Gamma_к(m, z(m))$ и $m^0 = \Gamma_к(m, \overline{z(m)})$. Для начальной вершины графа текущая маска содержит единицы. Итак, при входе в вершину m движение продолжается к вершине m^1 для тех разрядов i, где $M_1(i) = 1$, и к вершине m^0 – для тех разрядов, где $M_0(i) = 1$. Обработке подвергаются все вершины m, где $M(m) \neq 0$. При достижении висячих вершин $z(m)$ маска M_1 определяет разряды i, где $z_{к,i} = 1$, а M_0 – разряды i, где $z_{к,i} = 0$.

4. Понятие управляющих вершин

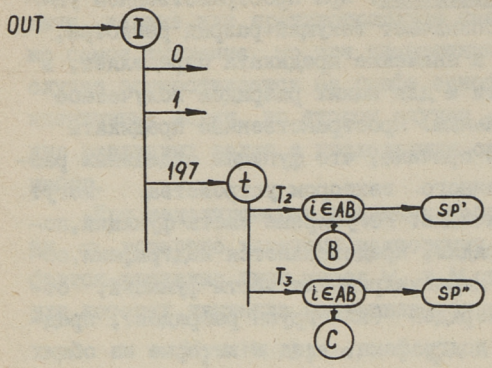
Вершины АГ делятся на два класса: функциональные и управляющие. Функциональные вершины взвешены переменными $z \in Z$, используемыми для описания функций ОД. Управляющие вершины взвешены предикатами и введены для управления движением на АГ в трех аспектах: в пространственном, временном и числовом. В зависимости от значения предиката, при движении по графу выбирается выход из управляющих вершин в один из двух подграфов для продолжения движения. Предикаты могут иметь вид $i * p$, где $* \in \{=, \neq, <, \leq, >, \geq, \in\}$, а p – целое число (или множество целых чисел). В случае, когда нужно перебирать большее число чем одно p, целесообразно заменить совокупность управляющих вершин одной, имеющей, соответственно, больше, чем два выхода.

Пространственное управление. При пространственном управлении переменная i обозначает текущий разряд регистра, представляемого графом, а значение предиката определяет, в какой подграф нужно войти и для каких разрядов получаемое решение будет действительным. Пространственные предикаты введены формально по той причине, что функции отдельных разрядов и полей представленного вектором устройства могут оказаться частично различными. Регулярные части функций, повторяющиеся во всех разрядах, представляются подграфами, общими для всех разрядов. Нерегулярные же части функций, отличные для отдельных разрядов (или групп разрядов), представляются собственными подграфами, вход в которые из общих подграфов происходит через управляющие вершины, взвешенные пространственными предикатами.

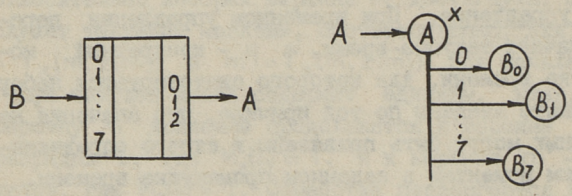
Временное управление. При временном управлении переменная i обозначает текущее время, а p - конкретный момент в промежутке времени, для которого синтезируется набор. Временные предикаты введены по той причине, что значения некоторых переменных могут быть привязаны к строго определенным фиксированным моментам в заданном промежутке времени. Так, например, если компонентом синтезируемого входного набора служит код команды, то этим набором охватывается промежуток времени, соответствующий циклу команды, но в то же время вводимые командой данные или выдаваемые ей результаты обычно привязаны к определенным тактам цикла.

Пример I. Модель команды PUSH B,C, имеющей код $11000101_2 = 197_{10}$, из системы команд микропроцессора Intel 8080 представлена на фиг. I. Здесь OUT - обобщенный выход МП, включающий адресную магистраль AB и магистраль данных DB как дополнение AB к OUT, I - целочисленная переменная, значениями которой служат коды команд, а B, C и SP - векторные переменные для обозначения регистров. Временные предикаты $t=T_2$, $t=T_3$ и пространственный предикат $i \in AB$ управляют вычислением значений OUT как во времени (в машинных циклах T_2 и T_3), так и по магистралям.

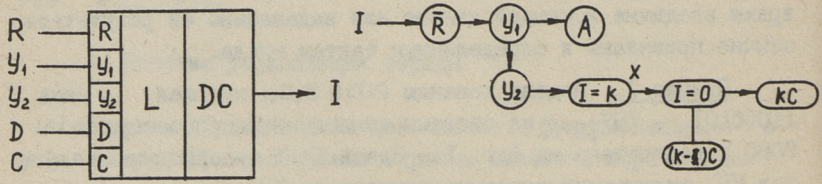
Числовое управление. При числовом управлении переменная i обозначает функцию, представляемую графом, а p - конкретное значение этой функции. Если значение предиката $i=p$ истинно, выходим из управляющей вершины в подграф G^p, за-



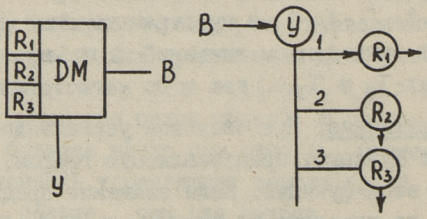
Фиг. 1.



Фиг. 2.



Фиг. 3.



Фиг. 4.

дающий условия, при выполнении которых функция, представленная графом, принимает значение p. Введение числовых предикатов позволяет сохранить двоичный характер выхода из графа даже для случая, когда функция, представляемая графом, не является двоичной. Так, например, при выходе из подграфа G^p в единичном направлении значением функции будет p. При выходе же из G^p в нулевом направлении значение функции не равно p. Введение в АГ механизма числового управления позволяет резко сократить перебор при синтезе наборов для установки определенных кодов (команд, адресов и т.д.). По необходимому значению p будет немедленно найден подграф, где могут быть найдены необходимые условия для установки $i = p$.

Пример 2. На фиг. 2 представлено функциональное обозначение шифратора, вырабатывающего 3-разрядный код ($S(A) = \{0,1,\ldots,7\}$) в зависимости от поступления двоичных сигналов $B_0, B_1, \ldots, B_7$, и соответствующая шифратору модель АГ. Числовое управление организовано при вершине A множеством предикатов A = 0, A = I, ..., A = 7.

Пример 3. На фиг. 3 представлена модель АГ для схемы, состоящей из счетчика L, реализующего микрооперации $\bar{R}$: L: = 0, y_1: L: = A, y_2: L: = L+1, и дешифратора D. Здесь $\bar{R}, y_1, y_2$ - простые двоичные переменные (сигналы управления), а L и A - векторные двоичные переменные. Числовым предикатом I = = k управляется решение установочной задачи для I (здесь k принимает некоторое желаемое целочисленное значение). В вершине I = 0 проверяется значение I в предыдущий момент времени. Последователи вершины I = 0 задают импульсную последовательность, необходимую для установки значения I: = k. Коэффициенты k или k − I определяют число импульсов в генерируемой последовательности.

5. Модель неисправностей и генерирование тестов

Традиционная модель константных неисправностей в вентильной схеме ОД может быть обобщена, используя предложенную модель АГ, на более высокий уровень - на модель неисправностей дешифрации, мультиплексирования и демультиплексирования, на модель ошибок операций и т.д. Формально эти дефекты приводимы к константным неисправностям на выходах вершин АГ.

Неисправности управляющей части ОД. Предлагаемая модель неисправностей базируется на предположении, что цифровая система делима на две части: на управляющий автомат (УА) и операционный автомат (ОА) [5]. Далее, относительно ОА может быть сделано предположение, что он реализован в виде М-автомата [5], имеющего магистральную структуру так, что все операции, реализуемые в ОА, распространимы на все его внутренние регистры. Такое предположение в общем случае приводит к избыточности во множестве предполагаемых дефектов управления ОА, что в результате приводит к избыточности получаемых тестов при сохранении их полноты. Введение такой избыточности необходимо, если информация о структуре ОА отсутствует.

Генерирование тестов на базе предложенной модели неисправностей направлено на проверку правильности работы выходов вершин АГ.

Рассмотрим вершину m, взвешенную целочисленной переменной y и имеющую n выходов (для значений $y = 1,2,\ldots,n$), входящих соответственно, в вершины $m_1, m_2, \ldots, m_n$. Обозначим событие $y = i$ через булеву переменную y_i, где

$$y_i = \begin{cases} 0, & \text{если } y \neq i, \\ 1, & \text{если } y = i. \end{cases}$$

Множество неисправностей при вершине m можно теперь представить следующим образом:

$$R(m) = \{ y_1 \equiv 0,\ y_1 \equiv 1, \ldots,\ y_n \equiv 0,\ y_n \equiv 1 \}.$$

Введем булеву переменную $l_{m,p}$, такую, что $l_{m,p} = 1$, если на графе активизирован некоторый путь из вершины m в вершину p, и $l_{m,p} = 0$, если при заданных значениях весовых переменных для вершин графа такого пути не существует. Заметим, что p может представлять и константу $p \in \{0,1\}$. Тогда под $l_{m,p} = 1$ понимается существование пути из m к одному из выходов графа.

Для проверки выхода вершин m, входящего в вершину m_i, необходимо генерировать тесты, выполняющие следующие два условия:

$$l_{н,m}\, l_{m_i,e} \bigwedge_{j,j\neq i} l_{m_j,\bar{e}} = 1, \quad e \in \{0,1\}, \qquad (4)$$

где $н$ – начальная вершина.

Легко убедиться, что таким образом составленные тесты всегда обнаруживают неисправность $y_i \equiv 0$, а также неисправности $y_i \equiv 1$, где $j \neq i$.

Тест, удовлетворяющий условию (4), проверяет следующие физические дефекты: отсутствие сигнала $y = i$ и появление дополнительно к $y = i$ других ложных сигналов $y = j$, где $j \neq i$.

Нетрудно понять, что тесты, построенные вышеописанным образом, обнаруживают все функциональные дефекты управления ОА, введенные, например, в [6], применительно к микропроцессорам.

Пример 4. Рассмотрим модель АГ, представленную на фиг. 4, для схемы, состоящей из трех регистров R_1, R_2, R_3 и мультиплексора. Функция управления мультиплексором задана на графе вершиной y, имеющей 3 выхода. Множество дефектов, возможных при управлении мультиплексором, имеет следующий вид:

$$R(y) = \{y_1 \equiv 0,\ y_1 \equiv 1,\ y_2 \equiv 0,\ y_2 \equiv 1,\ y_3 \equiv 0,\ y_3 \equiv 1\}.$$

Активизированные на графе пути, необходимые для теста, обнаруживающего неисправности $y_1 \equiv 0$, $y_2 \equiv 1$ и $y_3 \equiv 1$, показаны жирными стрелками. Тесту соответствует набор, где $y = I$, $R_1 = I$, $R_2 = 0$, $R_3 = 0$.

Заметим, что выполнение условий (4) происходит путем генерирования совокупности двоичных сигналов и кодов, представляющих собой значения целочисленных переменных. Двоичные сигналы от операционных элементов (например, регистров для хранения операндов и данных) можно, вообще говоря, интерпретировать как I-битовые операнды. Другими словами, предложенная методика приводит к синтезу тестов, различающих различные функции путем выполнения I-битовых операций. Легко понять, что при достаточно большом количестве и различаемых функций, т.е. при достаточно большом количестве выходов проверяемой вершины АГ, задача решения уравнения (4) путем генерирования I-битовых операндов, как правило, неразрешима. Выходом из положения является возможность выполнения условий (4) по частям на множестве нескольких I-битовых операндов.

Для проверки отсутствия дефектов управления по всей разрядной сетке ОА требуется циклическое проведение теста

на множестве сдвигаемых операндов так, чтобы все I-битовые операнды были использованы для проверки всех разрядов (для различения всех возможных ошибочных функций на всех разрядах).

Неисправности операционной части ОД. Для проверки операционной части ОД, в частности, функционального блока, при незнании его структуры требуется полный перебор. В присутствии информации о структуре оказывается возможным полный перебор относительно компонентов структуры. Так, например, если известно, что сумматор построен по структуре, реализующей последовательный перенос, а результаты операции во всех разрядах наблюдаемы, то может быть проведена независимая проверка всех разрядов сумматора методом полного перебора I-битовых операндов. Для установки нужных значений входных и внутренних переменных (в частности, переменных переноса), может быть использована также модель АГ.

6. Заключение

Основные результаты данной работы можно сформулировать следующим образом:

I. Предложена оригинальная графовая модель цифровых систем, позволяющая полностью формализировать процесс синтеза тест-программ.

2. Показано, что модель неисправностей формально вытекает из предложенной графовой модели и представляет собой обобщение классической модели константных неисправностей.

3. Расширение класса переменных, используемых в предложенной модели объекта, повышает эффективность процесса синтеза тестов в следующих аспектах: а) введение векторных двоичных переменных и переменных последовательностей импульсов позволяет распараллелировать процесс, соответственно, в пространственной и временной областях; б) введение целочисленных переменнных позволяет сократить перебор вариантов поиска решений.

Литература

I. Убар Р.Р. Генерирование тестов на модели альтернативных графов. - Тр. Таллинск. политехн. ин-та, № 409, 1976, с. 75-81.

2. A k e r s S.B. Binary Decision Diagrams. - IEEE Trans. on Computer, June 1978, vol. C-27, p. 509-516.

3. П л а к к М.П., У б а р Р.Р. Построение тестов цифровых схем на модели альтернативных графов. - Автоматика и телемеханика, 1980, № 5, с. 152-163.

4. U b a r R.R. Vektorielle alternative Graphen und Fehlerdiagnose für digitale Systeme. - Nachrichtenttechnik/ Elektronik, 1981, 31, H.I, S. 25-29.

5. М а й о р о в С.А., Н о в и к о в Г.И. Структура электронных вычислительных машин. Л., Машиностроение, 1979, с. 384.

6. T h a t t e S.M., A b r a h a m J.A. Test Generation for Microprocessers. - IEEE Trans. on Computer, June 1980, vol. C-29, p. 429-441.

R. Ubar

EIN VERALLGEMEINERTES MODELL VON ALTERNATIVEN GRAPHEN ZUR TESTSYNTHESE FÜR DIGITALE SYSTEME

Zusammenfassung

Ein originelles Graphenmodell der digitalen Systeme zur Automatisierung der Synthese von Testprogrammen wird vorgeschlagen. Neben der einfachen digitalen Variablen werden vektorielle und ganzzahlige Variablen sowie Folgevariablen eingeführt. Ein Fehlermodell als eine Verallgemeinerung des klassischen Modells von konstanten Fehlern wird abgeleitet.

№ 550

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED

ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

УДК 681.32

Р.Р. Убар

О СНИЖЕНИИ КОМБИНАТОРНЫХ ТРУДНОСТЕЙ ПРИ СИНТЕЗЕ ТЕСТОВ ДЛЯ ЦИФРОВЫХ АВТОМАТОВ

I. Введение

Трудности решения задач синтеза проверяющих тестов для цифровых схем связаны, в первую очередь, с размерностью модели объекта. Структурные методы синтеза тестов развивались, исходя из вентильного представления схемы. "Борьба" с размерностью модели привела к сжатию описания объекта путем суперпозиции функций вентилей и представлению деревообразных комбинационных подсхем одной функцией (в частности, в виде скобочной или нормальной форм). В результате сокращения общего числа переменных (тех, которые использовались для описания внутренних узлов в деревообразных подсхемах) размерность модели объекта в некоторой мере снизилась. Но проблема комбинаторики, связанная с перебором вариантов при поиске теста, сохранилась. Особую остроту имеет проблема перебора при синтезе тестов последовательностных схем, где перебору подвергаются не только варианты наборов для текущего такта, но и варианты последовательностей наборов в течение совокупности тактов.

В статье предлагается метод сжатия модели последовательностной схемы так, чтобы перебор вариантов решения был перенесен из пространственно-временной многотактовой области в пространственную однотактовую область.

2. Принцип одновременного синтеза групп сигналов

Ниже будет использован принцип синтеза тестов, где наравне с шагами фиксации одиночных двоичных сигналов возможны шаги одновременной фиксации нескольких сигналов, в част-

ности, целой функциональной группы сигналов. Здесь возможны два случая:

I) группа сигналов представляет собой состояние регистра (например, регистра операционного автомата (ОА), предназначенного для хранения данных), внешняя установка которого тривиальна;

2) группа сигналов представляет собой состояние памяти управляющего автомата (УА), внешняя установка которого происходит через другие состояния.

В последнем случае возможны две ситуации:

I) закономерности перехода от состояния к состоянию заданы явно, например, в виде микропрограммы;

2) закономерности этих переходов заданы неявно, например, в спрятанном виде в принципиальной схеме автомата.

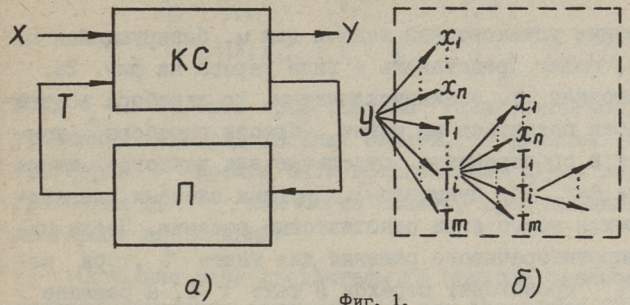
Сущность предлагаемого принципа заключается во введении в структурную модель объекта информации о явных закономерностях перехода между состояниями и использовании ее с целью сокращения числа переборов при синтезе тестов.

Рассмотрим цифровой автомат, представленный на фиг. Iа, состоящий из комбинационной схемы КС и памяти П, функции которого могут быть выражены в следующем виде:

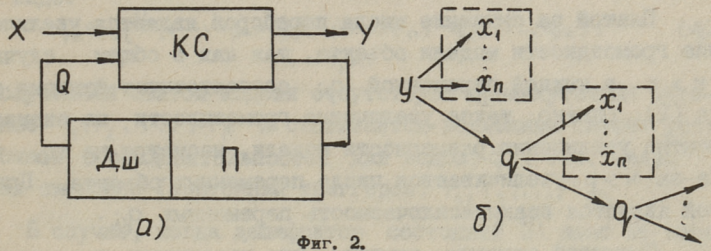
$$y = f(x_1, \ldots, x_n;\ T_1, \ldots, T_p). \qquad (I)$$

Процесс решения установочной задачи для y можно представить в виде дерева на фиг. Iб, где каждая фиксация значения переменной в общем случае приводит к следующим установочным задачам. Легко заметить, что процесс генерирования установочных задач распространяется не только в пространстве (т.е. на новые узлы схемы, заданные переменными $x_i \in X$ и $T_j \in T$), но и через триггеры T_j и во времени (т.е. предыдущие такты). Так как противоречивые фиксации значений переменных в общем случае непрогнозируемы, то неизбежным становится перебор вариантов при поиске решения. Областью перебора является как пространство, так и время (обведена на фиг. Iб пунктиром).

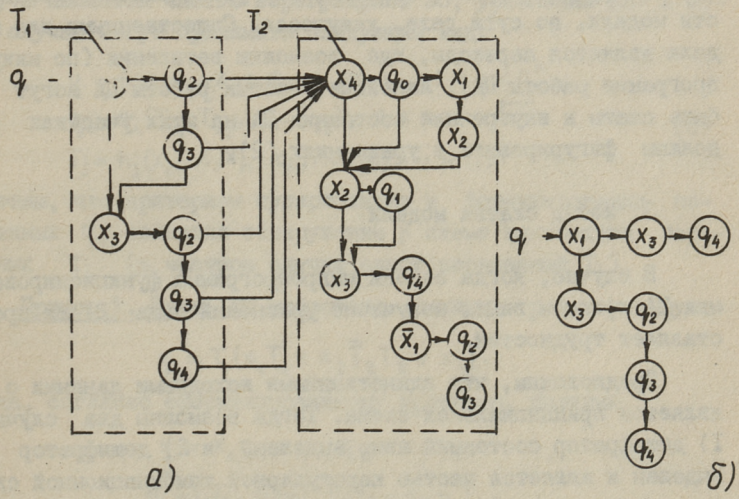
Если представить цифровой автомат (ЦА) в виде, где дешифратор Дш состояний автомата выделен из КС (фиг. 2а), то возможно приведение функции (I) к следующему виду:



Фиг. 1.



Фиг. 2.



Фиг. 3.

$$y = f(x_1, \ldots, x_n;\ q_1, \ldots, q_k). \qquad (2)$$

Процесс решения установочной задачи для y, базирующийся на функции (2), можно представить в виде дерева на фиг. 2б. Так как состояния q_i взаимоисключаемы, то перебора во временной области появиться не может. Области перебора концентрируются в пространстве, отдельные для каждого такта (обведены на фиг. 2б пунктиром). Другими словами, перебору подвергаются всего лишь однотактовые решения. После нахождения непротиворечивого решения для такта t, при необходимости, происходит переход в такт $t-1$, а решение для такта t изменяться больше не будет.

Платой за снижение числа переборов является увеличение громоздкости модели объекта, так как в общем случае $k > p$, а каждой переменной q_i соответствует функция вида (2). Однако такое увеличение громоздкости не эквивалентно увеличению размерности модели, несмотря на то, что из-за $k > p$ увеличивается число переменных объекта. Причиной является взаимоисключаемость переменных q_i.

С другой стороны, заметим, что увеличение громоздкости модели, по сути дела, кажущееся. Существенными для модели являются переходы, где возможны ветвления (по микропрограмме работы ЦА). Линейные участки работы ЦА могут быть сжаты и внутренние состояния ЦА на этих участках не должны фигурировать в уравнениях (2).

3. Метод сжатия модели

В случае, когда задана микропрограмма функционирования ЦА в явном виде, получение уравнений вида (2) не представляет трудностей.

Предположим, что единственными исходными данными о ЦА является принципиальная схема. Тогда возможны два случая: 1) дешифратор состояний явно выделен, и 2) дешифратор не выделен и является частью нерегулярной комбинационной схемы.

Рассмотрим сначала первый случай как более простой. Обычными методами суперпозиции элементов схемы можно генерировать следующие уравнения непосредственных связей:

$$
\begin{aligned}
&y = f(x_1,\ldots,x_n;\ q_1,\ldots,q_k)\\
&q_i = g_i(T_1,\ldots,T_p),\quad i=\overline{1,k};\\
&T_j = h_j(x_1,\ldots,x_n;\ q_1,\ldots,q_k),\quad j=\overline{1,p}.
\end{aligned}
\qquad (3)
$$

Если y может быть представлена также системой нескольких уравнений непосредственных связей, то функция возбуждения триггеров T_j должна быть представлена одним выражением. Уравнения для q_i, как легко понять, представляют собой элементарные конъюнкции.

Основная идея предлагаемого подхода заключается в продолжении операции суперпозиции для уравнений (3) в следующем виде:

$$
q_i = g_i(h_1,\ldots,h_p) = g_i(x_1,\ldots,x_n;\ q_1,\ldots,q_k),\quad i=\overline{1,k}. \qquad (4)
$$

В полученной сжатой модели отсутствуют промежуточные переменные $T_1, T_2, \ldots, T_p$ и оказывается возможным синтез установочных последовательностей для кодов q_i, т.е. целых групп сигналов (состояний триггеров) $T_1, T_2, \ldots, T_p$.

В случае, когда дешифратор состояний ЦА явно не выделен, а является частью нерегулярной КС, уравнения (3) в общем случае могут принимать следующий вид:

$$
\begin{aligned}
&y = f(x_1,\ldots,x_n;\ q_1,\ldots,q_v),\quad 1<v<2^k;\\
&q_i = g_i(T_{i_1},T_{i_2},\ldots,T_{i_s}),\quad s\le p,\ i=\overline{1,v};\\
&T_j = h_j(x_1,\ldots,x_n;\ q_1,\ldots,q_v),\quad j=\overline{1,p}.
\end{aligned}
\qquad (5)
$$

Заметим, что критерием суперпозиции и формулирования переменных q_i является обнаружение в схеме конъюнкций переменных T_j (в частном случае одной переменной T_j).

Пример I. Пусть КС цифрового автомата реализует функцию

$$
y = T_1(x_1T_2 \vee x_2\bar{T}_2T_3 \vee x_3).
$$

Тогда, согласно сформулированному критерию, получим

где

$$
y = x_1q_1 \vee x_2q_2 \vee x_3q_3,
$$

$$
q_1 = T_1T_2,\ q_2 = T_1\bar{T}_2T_3,\ q_3 = T_1.
$$

Нетрудно заметить, что в уравнениях (5) переменные становятся невзаимоисключающими. Выделим следующие возможные виды отношений между переменными q_i и q_j:

$$q_i \cap q_j = \emptyset, \quad (\text{Пример: } q_i = T_1 T_2 T_3,\ q_j = T_1 T_2 \bar{T}_3),$$
$$q_i \cap q_j \neq \emptyset, \quad (\text{Пример: } q_i = T_1,\ q_j = T_2), \qquad (6)$$
$$q_i \subset q_j \quad (\text{Пример: } q_i = T_1 T_2 T_3,\ q_j = T_1 T_2).$$

Факт возможности невзаимоисключения переменных q_i и q_j влияет на предлагаемый метод в следующих аспектах:

I) при сжатии модели (сокращении внутренних переменных $T_1, \ldots, T_m$) усложняется процесс сокращения получаемых при суперпозиции уравнений вида (4);

2) при синтезе тестов усложняется процесс генерирования установочной задачи для предыдущего такта.

4. Генерирование уравнений перехода

Процесс генерирования уравнений перехода путем суперпозиции функций возбуждения триггеров в виде (4) может быть проведен на модели АГ [I]. Преимуществом этой модели по сравнению с другими видами задания булевых функций является простота преобразования графов с целью упрощения результата суперпозиции.

Рассмотрим модель функции возбуждения $T_j = h_j(x_1, \ldots, x_n; q_1, \ldots, q_k) = h_j(X, Q)$ в виде альтернативного графа $G_{T,j}(\Gamma_{T,j}, M_{T,j})$, где $M_{T,j}$ – множество вершин, взвешенных переменными $z \in X \cup Q$, а $\Gamma_{T,j}$ – отображение множества $M_{T,j}$ в множество $M_{T,j}$. Здесь $\Gamma_{T,j}(m)$ – множество последователей вершины $m \in M_{T,j}$. Так как функции $q_i = g_i(T_1, \ldots, T_m)$ представляют собой простые конъюнкции, то процесс суперпозиции (4), согласно определению АГ [I], заключается в синтезе некоторого нового графа $G_{q,i}$ путем простого объединения графов $G_{T,j}$ так, чтобы появились следующие дополнительные связи:

$$\forall_{\kappa,m} : \Gamma_{q,i}(m) = m^{H}_{\kappa+1},$$
$$\kappa = \overline{1, p-1}, \quad m \in \{m \mid \Gamma_{T,\kappa}(m,1) = \phi\}, \qquad (7)$$

где $m^{H}_{\kappa+1} \in M_{T,\kappa+1}$ – начальная вершина в графе $G_{T,\kappa+1}$, а $\Gamma_{T,\kappa}(m,1)$ – первый последователь вершины m в графе $G_{T,\kappa}$.

Пример 2. Рассмотрим в качестве примера функции возбуждения двух D-триггеров:

$$T_1 = x_1(q_2 \vee q_3) \vee x_3(q_2 \vee q_3 \vee q_4),$$

$$T_2 = x_4 q_0 (x_1 \vee x_2) \vee x_2 q_1 \vee x_3(q_4 \vee \bar{x}_1(q_2 \vee q_3)),$$

представленные в виде модели АГ на фиг. 3 и объединенные, согласно правилу (7), в граф, изображающий функцию $q_y = T_1 T_2$.

В случае, когда функции T_j встречаются в конъюнкции в инвертированном виде $\bar{T}_j$, необходимо вместо графа $G_{T,j}$ взять инвертированный граф $\bar{G}_{T,j}$ [I].

5. Упрощение альтернативных графов

Введем теперь следующий метод упрощения графа, полученного по алгоритму (7). Метод заключается в прохождении всех возможных путей в синтезированном графе и в копировании их в виде дерева. При этом возможны два вида сокращений вершин:

I) Сокращение в прямом движении: вершина m, весовая переменная которой $z(m)$ ранее встречалась, не копируется, а направление дальнейшего движения определяется значением $z(m)$, которое зафиксировалось при первой ее встрече в некоторой вершине m' (т.е. при выборе выхода из m').

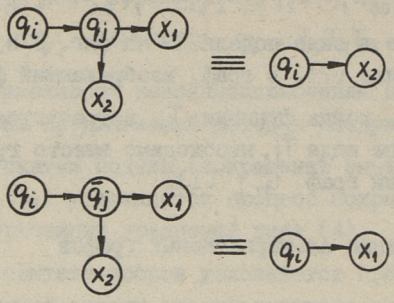
Замечание I. Из-за взаимоисключения переменных $q_i \in Q$, всегда копируется вершина первой встречи любого из $q_i \in Q$ (при встрече следующих $q_j \in Q$, принимается $q_j = 0$).

Замечание 2. В случае, когда $q_i \in Q$ невзаимоисключаемы, требуется проведение учета значений переменных $T_1, \ldots, T_p$ при встрече каждого $q_i \in Q$ с целью принятия решения о значении ее и о необходимости копирования ее.

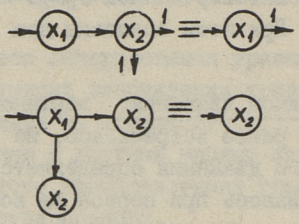
Некоторые примеры сокращения вершин при условии взаимоисключения переменных приведены на фиг. 4.

2) Сокращение в обратном движении возможно по двум способам:

а) если для некоторой вершины m в копированном графе можно установить, что оба выхода из него приводят к одному и тому же выходу $e \in \{0,1\}$, то она исключается из графа, а значение e присваивается входу этой вершины с целью продолжения анализа в обратном направлении;



Фиг. 4



Фиг. 5.

б) если для некоторой вершины m в копированном графе по обоим выходам образуется вход в эквивалентные подграфы, то вершина m и один из эквивалентных подграфов удаляются, а вход в вершину m становится входом в другой эквивалентный подграф.

Некоторые примеры сокращения копированного графа в обратном движении иллюстрируются на фиг. 5, а фиг. 3б представляет сокращенный по изложенному методу граф q_1, показанный на фиг. 3а.

6. Заключение

Предложен новый способ сжатия размерности модели цифровых автоматов, базирующийся на применении метода суперпозиции булевых функций.

В получаемой новой модели вместо некоторого множества внутренних сильносвязанных переменных введено новое множество слабосвязанных переменных, которые в частном случае можно интерпретировать как переменные состояния объекта.

Предложенный метод приводит к возможности одновременного генерирования групп взаимосвязанных сигналов, что в результате резко сокращает перебор вариантов при поиске теста.

Литература

I. Плакк М.П., Убар Р.Р. Построение тестов цифровых схем на модели альтернативных графов.- Автоматика и телемеханика, 1980, № 5, с. 152-163.

R. Ubar

ZUR REDUZIERUNG VON KOMBINATORISCHEN SCHWIERIGKEITEN BEI DER TESTERSTELLUNG FÜR DIGITALE AUTOMATEN

Zusammenfassung

Es wird ein Verfahren zur Verdichtung des Modells für digitale Automaten vorgeschlagen. Auf Grund der gegebenen Funktionen des Objekts werden neue Variablen und Funktionen eingeführt, die eine parallele Verarbeitung von mehreren dichtgekoppelten Signalen zuläßt. Als Ergebnis reduzieren sich die Schwierigkeiten der Testerstellung - eine Aufgabe der Synthese von Belegungsfolgen wird mit einer Menge von Aufgaben der Synthese von einfachen Belegungen ersetzt.

№ 550

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED

ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

УДК 681.32

А.А. Вийлуп, Т.А. Эвартсон

ГЕНЕРИРОВАНИЕ ТЕСТОВ ДЛЯ МИКРОПРОЦЕССОРНЫХ БИС

Одной из сложнейших задач, стоящих перед изготовителями микропроцессорных БИС, а также устройств, реализованных на микропроцессорных наборах БИС, является обеспечение достоверного контроля работоспособности изделий. Высокая степень интеграции и ограниченное количество контрольных точек в таких БИС делают невозможным применение регулярных методов синтеза тестов, основанных на вентильном представлении контролируемой схемы.

В статье рассматривается метод формального генерирования тест-программ для микропроцессорных БИС, основанный на применении обобщенной модели АГ [1]. Модель обеспечивает сжатое описание функций объекта диагностирования (ОД), а также формальное задание неисправностей на более высоком уровне, чем логические константы в узлах вентильных схем.

Модель объекта диагностирования. Представим ОД в виде множества функций

$$z_к^t = f_к(Z_к^{t-1}), \quad z_к \in Z, \quad Z_к \subseteq Z, \qquad (1)$$

описывающих поведение компонентов ОД на микропрограммном или программном уровнях и представленных в виде обобщенных АГ $G_к=(M_к,\Gamma_к)$, где $M_к$ - множество вершин АГ, взвешенных переменными $z \in Z_к$, а Z - множество всех переменных, используемых для описания ОД. Отображение $\Gamma_к(m)$ задает множество последователей вершины m. Каждая переменная $z \in Z$ задана своей областью определения $S(z)$, причем z могут представлять логические потенциалы и импульсы, логические векторы, коды, импульсные последовательности и т.д.

Пример 1. Рассмотрим модель АГ, представленную на фиг. 1 и описывающую подмножество списка команд микропроцессора Intel 8080, приведенное в табл. 1.

Табл. I

I_1: MVI A,D	(A) ← IN2
I_2: MOV RI,A	(R) ← (A)
I_3: MOV R2,A	(R2) ← (A)
I_4: MOV R3,A	(R3) ← (A)
I_5: MOV M,RI	((R2),(R3)) ← RI
I_6: MOV M,A	((R2),(R3)) ← A
I_7: MOV RI,M	(RI) ← ((R2),(R3))
I_8: MOV A,M	(A) ← ((R2),(R3))
I_9: ADD RI	(A) ← (A) + (RI)
I_{10}: ORA RI	(A) ← (A) ∨ (RI)
I_{11}: ANA RI	(A) ← (A) ∧ (RI)
I_{12}: CMA	(A) ← ($\overline{A}$)
I_{13}: IMP	(PC) ← (IN3)(IN2)
I_{14}: JMPX	(PC) ← (IN3)(IN2), if cond. X=I
I_{15}: CALL M	(SP)-I) ← (PCH), if t=t_2
	((SP)-2) ← (PCL), if t=t_3
	(SP) ← (SP) - 2
	(PC) ← (IN3)(IN2)
I_{16}: RET	(PCL) ← ((SP)), if t=t_2
	(PCH) ← ((SP)+I), if t=t_3
	(SP) ← (SP) + 2
I_{17}: PUSH	((SP)-I) ← (R2), if t=t_2
	((SP)-2) ← (R3), if t=t_3
	(SP) ← (SP) - 2
I_{18}: POP	(R3) ← ((SP)), if t=t_2
	(R2) ← ((SP)+I), if t=t_3
	(SP) ← (SP)+2
I_{19}: SPR	(SP) ← (R2),(R3)

Табл. 2

I_{18}: POP	(I=18)
IN2 = D_I	
IN3 = D_2	
I_7: MOV RI,M	(I=7)
IN2 = D_3	
I_1: MVI A,D	(I=1)
IN2 = D_4	
I_9: ADD RI	(I=9)
I_6: MOV M,A	(I=6)
t_2: OUT(DB) = D_3 + D_4	
OUT(AB) = D_2,D_I	

Табл. 3

Операнды	A = D_5	I	I
	RI = D_4	I	I
Результаты	I = 9	I	0
	I = I0		I
	I = II		I
	I = I2	0	0

Здесь A - аккумулятор, R1,R2 и R3 образуют блок регистров общего назначения, PC - счетчик команд, SP - указатель стека, а IN и OUT представляют, соответственно, вход и выход микропроцессора относительно окружающего мира. Переменные IN2 и IN3, соответственно, обозначают 2- и 3-й байты команды. Символы H и L введены для обозначения старшего и младшего байтов рассматриваемой векторной переменной, а t_2 и t_3 обозначают второй и третий машинный цикл (состояния микропроцессора, при которых или результаты теста должны быть выведены из микропроцессора, или входные наборы (операнды) должны быть введены в микропроцессор).

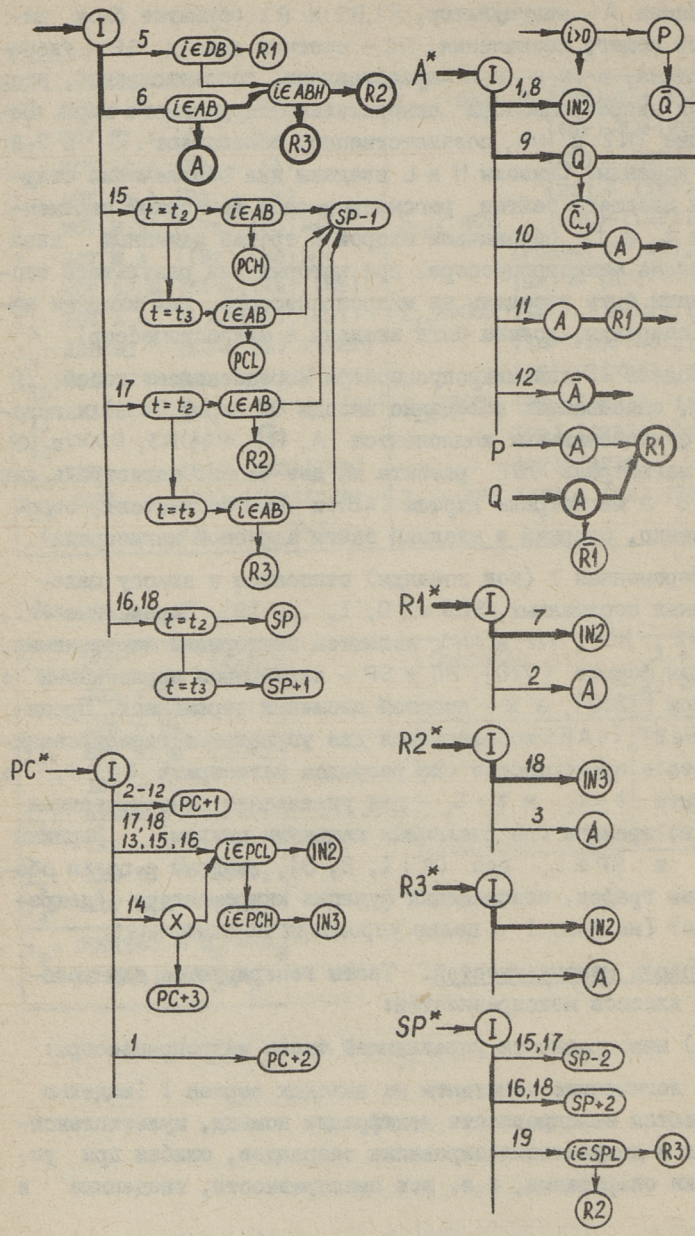
Модель АГ для микропроцессора представляет собой IO графов, описывающих поведение выхода ОД (граф OUT) и поведение функциональных компонентов A, R1, R2, R3, PC и SP. Здесь магистраль OUT разбита на две части: магистраль данных DB и магистраль адреса (ABH и ABL обозначают, соответственно, старший и младший байты адресной магистрали).

Переменная I (код команды) относится к классу целочисленных переменных, $S(I) = \{0, I, \ldots I9\}$. Переменные A, R1, R2, R3, IN2 и IN3 являются векторными переменными, имеющими формат {7:0}, PC и SP - векторными переменными с форматом {I5:0}, а x - простой двоичной переменной. Предикаты $i \in DB, i \in AB$ используются для управления генерированием теста в пространстве (по разрядам магистрали OUT), а предикаты $t = t_2$ и $t = t_3$ - для управления генерированием теста во времени (по различным машинным циклам). Вершины $PC \pm i$ и $SP \pm i$, где $i \in \{I, 2, 3\}$, введены с целью обозначения графов, описывающих функции инкрементора (декрементора) (на фиг. I с целью упрощения не показаны).

Модель неисправностей. Тесты генерируются для следующих классов неисправностей:

I) неисправности управляющей части микропроцессора:

- логические константы на выходах вершин I (моделью покрываются неисправности дешифрации команд, мультиплексирования и демультиплексирования операндов, ошибки при управлении операциями, т.е. все неисправности, введенные в [2];


I
5
i∈DB
R1
6
i∈AB
i∈ABH
R2
R3
A
15
t=t2
i∈AB
SP-1
PCH
t=t3
i∈AB
PCL
17
t=t2
i∈AB
R2
t=t3
i∈AB
R3
16,18
t=t2
SP
t=t3
SP+1
A*
I
i>0
P
Q̄
C-1
1,8
IN2
9
Q
C̄-1
10
A
11
A
R1
12
Ā
P
A
R1
Q
A
R̄1
R1*
I
7
IN2
2
A
R2*
I
18
IN3
3
A
R3*
I
IN2
A
SP*
I
15,17
SP-2
16,18
SP+2
19
i∈SPL
R3
R2
PC*
I
2-12
PC+1
17,18
13,15,16
i∈PCL
IN2
14
X
i∈PCH
IN3
PC+3
1
PC+2

Фиг. 1.

2) неисправности операционной части микропроцессора:

- логические константы на выходах вершин, весовыми переменными которых являются двоичные переменные и векторные переменные (моделью покрываются логические константы на выходах регистров и магистралей);

- короткие замыкания между шинами и регистрами, а также между разрядами шин и регистров;

- функциональные неисправности в операционных элементах (в регистрах, счетчиках, АЛУ и т.д.).

Разобьем множество R всех возможных неисправностей на 2 класса:

$$R = R_{АГ} \cup R_{ЛГ}. \qquad (2)$$

Здесь $R_{АГ}$ - класс неисправностей логических констант, задаваемых непосредственно моделью АГ;

$R_{ЛГ}$ - класс неисправностей КЗ между разрядами и функциональных неисправностей в операционных элементах, требующих для проверки специальных локальных тестов (или дополнительных условий [3]).

Генерирование тестов. Генерирование тестов для микропроцессорных систем включает следующие этапы:

- синтез последовательности команд (тест-программы), предназначенных для загрузки и транспортировки нужных операндов, для проведения проверяемой операции и для транспортировки результатов теста к выходу ОД;

- синтез таких операндов, чтобы удовлетворялись условия для обнаружения заданных дефектов.

На модели АГ задача синтеза тестов формулируется как задача синтеза тестов для проверки вершин АГ. Вершины, подлежащие проверке, подбираются в процессе последовательной активизации путей, проводимом так, чтобы все вершины в графах оказались хотя бы один раз пройденными.

Пусть $l_{н,e}$ - текущий активизированный путь в графе $G_к$ из начальной вершины $н$ к выходу e графа, где $e \in \{0,1\}$, проходящий некоторое подмножество вершин $M'_к \subseteq M_к$. Если существует хотя бы одна непроверенная вершина $m \in M_к$, для нее строится тест.

Для проверки неисправностей $r \in R_{АГ}$ создаются тесты для проверки всех выходов вершины m. Пусть вершина m взвешена переменной $z(m) \in Z$, где $S[z(m)] = \{1,2,\ldots,n\}$, а каждому значению $z(m) = i$, $i \in S[z(m)]$ соответствует дуга в графе из вершины m в вершину m_i. В частном случае несколько различных значений $z(m)$ могут соответствовать одному и тому же последователю m_i.

Обозначим событие $z(m)=i$ через булеву переменную y_i, где

$$y_i = \begin{cases} 0, \text{ если } z(m) \neq i, \\ 1, \text{ если } z(m) = i. \end{cases} \qquad (3)$$

Тогда множество дефектов при вершине m, относящихся к классу $R_{АГ}$, принимает вид

$$R(m) = \{ y_i \equiv 0,\ y_i \equiv 1 \mid i = \overline{1,n} \}. \qquad (4)$$

Неисправности $y_i \equiv 0$ проверяются тестами, обеспечивающими выполнение условий

$$l_{н,m}\, l_{m_i,e} \bigwedge_{j=\overline{1,n};\, j \neq i} l_{m_j,\bar{e}} = 1,\ e \in \{0,1\}, \qquad (5)$$

а неисправности $y_i \equiv 1$ – тестами, при которых выполнены условия

$$l_{н,m}\, l_{m_j,e} \bigwedge_{i=\overline{1,n};\, i \neq j} l_{m_i,\bar{e}} = 1,\ e \in \{0,1\}. \qquad (6)$$

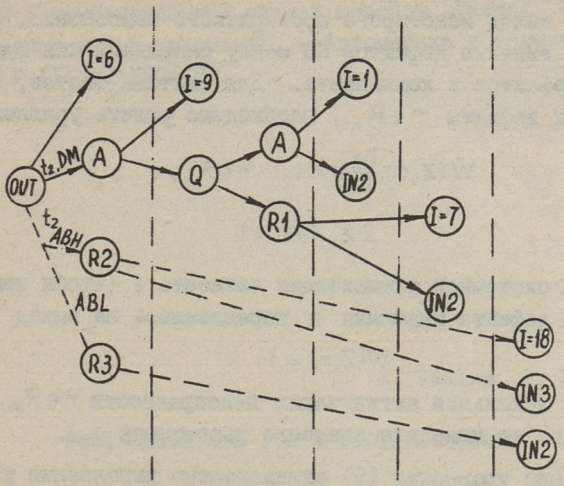
Исходя из вышесказанного, для проверки вершины m требуется множество тестов, при которых выполняются условия

$$l_{н,m}\, l_{m_i,e} \bigwedge_{j=\overline{1,n};\, j \neq i} l_{m_j,\bar{e}} = 1,\ e \in \{0,1\}, \qquad (7)$$

для всех $i = \overline{1,n}$.

В результате активизации путей $l_{н,m}$, $l_{m_i,e}$ и $l_{m_j,\bar{e}}$, где $j = \overline{1,n}$, $j \neq i$, согласно уравнению (7), генерируются тест-программы и совокупность операндов, проверяющих неисправности $r \in R_{АГ}$ при вершине m.

Пример 2. Рассмотрим процесс генерирования теста для проверки команды ADD R1. в примере I. На модели АГ (фиг. I) для этого необходимо генерировать тест для проверки вершины I в графе A при значении I = 9. Процесс генерирова-



Фиг. 2.

ния соответствующей тест-программы иллюстрируется на фиг.2 в виде дерева D , где каждой невисячей вершине d_K соответствует некоторый АГ G_K, а последователям d_K - вершины, пройденные в графе G_K. Висячим вершинам дерева соответствуют команды, составляющие тест-программу, а также используемые в программе операнды. Пунктирными линиями иллюстрирован процесс дополнения тест-программы для увеличения числа проверяемых неисправностей (при выполнении команды MOV,M,A (I = 6) возможна не только транспортировка результата теста, а также и наблюдение состояний регистров R2 и R3). Генерированная на дереве D (фиг. 2) тест-программа приведена в табл. 2. Генерирование операндов D_3 и D_4 для проверки выхода I = 9 вершины I осуществляется путем активации множества путей на графе A таких, чтобы выход из вершины I при I = I0, I = II и I = I2 привел к выходу из графа A в другом направлении, чем привел бы выход из вершины I в случае I = 9. Найденные 2 однобитовых операнда приведены в табл. 3. Все активизированные при синтезе теста пути показаны на модели АГ (фиг. I) жирными стрелками.

Для проверки неисправностей $r \in R_{AT}$ при вершине m необходимо использовать локальные тесты.

Обозначим через $z_к$ некоторый наблюдаемый выход ОД, а через z – выход некоторого проверяемого компонента, подвергаемый влиянию дефектов КЗ между разрядами или функциональных дефектов в компоненте. Для синтеза тестов, обнаруживающих дефекты $r \in R_{AT}$, необходимо решить уравнения

$$W(Z,r)\frac{\partial z_к}{\partial z}=1, \quad r \in R_{AT}, \tag{8}$$

где

$$\partial z_к/\partial z = 1 \tag{9}$$

– условие системной активизации элемента z (чтобы изменение из-за дефекта значения z передавалось на выход $z_к$), а

$$W(Z,r)=1 \tag{10}$$

– условие локальной активизации неисправности $r \in R_{AT}$ (чтобы присутствие ее изменило значение переменной z).

Решение уравнения (9) эквивалентно выполнению условия

$$l_{н,m} l_{m_1,e} l_{m_0,\bar{e}} = 1, \quad e \in \{0,1\}, \tag{11}$$

где $z \equiv z(m)$, а m_1 и m_0 – последователи вершины m соответственно, при $z = 1$ и $z = 0$. Дополнительными условиями $W(Z,r)=1$ могут служить или известные заранее тесты для компонента ОД, выходом которого является z, тесты, синтезируемые по структуре компонента (если она известна), или тестами, синтезируемыми по методу полного перебора.

Заключение

Рассмотренный метод синтеза тестов отличается простотой и открывает возможности для полной автоматизации диагностирования микропроцессорных БИС. В основе метода лежит обобщенная модель АГ, дающая сжатое описание функций объекта, а также возможность моделирования неисправностей на более высоком уровне, чем вентильные схемы.

Литература

I. Убар Р.Р. Обобщенная модель альтернативных графов для синтеза тестов цифровых систем. См. наст. сб. с.97.

2. Thattes S.M., Abraham J.A. Test generation for microprocessors. – IEEE Trans. on Computer, 1980, June, vol. C-99, p. 429-441.

3. У б а р Р.Р. Выделение подозреваемых неисправностей в комбинационных схемах методом решения булевых дифференциальных уравнений. - Автоматика и телемеханика, 1979, № II, с. 170-183.

A. Viilup, T. Evartson

TESTERSTELLUNG FÜR MIKROPROZESSOREN

Zusammenfassung

Eine Methode zur formalen Synthese von Testprogrammen zur Prüfung von Mikroprozessoren wird betrachtet. Die Methode beruht auf dem verallgemeinerten Modell von alternativen Graphen. Ein Fehlermodell wird eingeführt, das sich direkt aus dem Graphenmodell ergibt. Der Prozeß der Testerstellung wird in zwei Teilen durchgeführt: die Synthese von Befehlfolgen und die Synthese von Operanden. Beide Aufgaben werden mit dem Modell von alternativen Graphen gelöst. Algorithmen und Beispiele werden behandelt.

Содержание

I. О.А. Аарна. Математическая модель непрерывного автоматизированного технологического комплекса и ее применение в АСУ ТП..... 3

2. О.А. Аарна. Декомпозиция задачи оценки состояния химико-технологической системы..... I5

3. Ю.Ю. Ребане. Генератор модулей аппаратов химико-технологических схем..... 27

4. М.В. Мин. Динамический синтез высокоточных следящих систем..... 35

5. К.А. Райенд. Декомпозиция линейной модели объекта управления..... 4I

6. А.А. Кийтам, Э.К. Сакс. Приближенная оптимизация алгоритмов компенсационного управления с режимными ограничениями для многостадийных технологических процессов..... 55

7. Е.А. Венделин. Идентификация объектов при планировании вычислений..... 67

8. П.Л. Лейс, К.Х. Салум. Итеративный метод декомпозиции конечных автоматов..... 8I

9. А.Э. Кеэваллик, П.А. Китсник, А.В. Судницын. Метод программной реализации микропрограммного автомата..... 9I

I0. Р.Р. Убар. Обобщенная модель альтернативных графов для синтеза тестов цифровых систем..... 97

II. Р.Р. Убар. О снижении комбинаторных трудностей при синтезе тестов для цифровых автоматов..... III

I2. А.А. Вийлуп, Т.А. Эвартсон. Генерирование тестов для микропроцессорных БИС..... I2I

Teadusliku raamatukogu
ENSV TA
TALLINN
АН ЭССР
Научная библиотека

ТАЛЛИНСКИЙ ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ
Труды ТПИ № 550
РАСЧЕТ И ПРОЕКТИРОВАНИЕ СИСТЕМ
ТЕХНИЧЕСКОЙ КИБЕРНЕТИКИ

Электротехника и автоматика ХХП

Редактор Э. Рюстерн. Техн. редактор М. Тамме
Сборник утвержден коллегией Трудов ТПИ 01.04.83

Подписано к печати 25 октября 1983 года.
Формат 60x90/16. Печ. л. 8,25 + 0,5 приложение.
Уч.-изд. л. 7,0. Тираж 350. МВ-06170.
Ротапринт ТПИ, Таллин, ул. Коскла, 2/9. Зак. № 691.
Цена 1 руб. 05 коп.

руб. 1.05