# АНАЛИЗ РАЗМЕРНОСТИ В АЛГОРИТМИЧЕСКИХ СЕТЯХ

**Д. С. Васильченко, В. Е. Марлей (Санкт-Петербург)**

Анализ практической работы с различными категориями пользователей при разработке и эксплуатации математических моделей в различных предметных областях позволил выявить некоторые общие закономерности в мышлении и восприятии проблем:

– дискретность описания и восприятия анализируемого процесса, стремление рассматривать процесс, с шагом, равным привычному дискрету наблюдения и/или съема информации;

– стремление представить анализируемый процесс как некоторый связный сценарий, с описанием пред- и постусловий выполнения каждого этапа;

– стремление свести описание к некоторой регулярной циклически повторяющейся процедуре;

– стремление рассматривать этапы сценария в основном во временной последовательности;

– стихийный параллелизм описания сценария, заключающийся в частом использовании связок типа “а в это время”;

– стремление уменьшить число используемых единиц измерения и число анализируемых показателей и нежелание использовать непривычные единицы измерения и показатели;

– в большинстве случаев довольно четкое представление о допустимых и оптимальных границах значений анализируемых показателей;

– недоверие к результатам, полученным методами, которые они не могут наложить на свой сценарий процесса;

– частое стихийное сведение анализируемого процесса к описанию взаимодействия некоторых “хранилищ”, “емкостей” (например, какого-либо ресурса, результирующего продукта и т.п.).

Исходя из перечисленных закономерностей, можно составить следующий сценарий формирования модели анализируемого процесса пользователем, не слишком обремененным математической подготовкой. Первоначально окружающий мир воспринимается дискретно и локально. Человек фиксирует состояние интересующей его части мира, затем производит на него воздействие или фиксирует какое-либо производимое в это время воздействие и затем фиксирует новое состояние, воздействие фиксируется как причина изменения состояния, новое состояние – как следствие воздействия. Имеется стремление свести создаваемую модель к некоторой циклически повторяемой регулярной процедуре. Используемое стихийно человеком описание анализируемого процесса, выполняемое как последовательность выделенных этапов, существенно отличается своим “естественным” параллелизмом. Человек описывает процесс по этапам в естественной дискретности, не определяемой требованиями какого-либо метода вычислений. Если он наблюдает и воспринимает процесс с дискретом сутки, то и модель, сложившаяся в его мозгу, будет иметь дискрет сутки и предсказывать состояние мира на следующие сутки; если процесс определяется не по времени, а, например, по расстоянию и дискрет отметки этапа километр, то и модель имеет дискрет километр. При этом есть подсознательное стремление свести все варианты отметки этапа к временным. Доверие к реализуемой модели повышается по мере включения в нее все большей части известных из практического опыта закономерностей в виде, который позволяет пользователю их распознать.

В дальнейшем будем ориентироваться на способ представления моделей, позволяющий описать моделируемый процесс как регулярную циклически повторяющуюся процедуру, с учетом дискретности восприятия и естественного параллелизма и позволяющую пользователю сопоставить вычислительную структуру модели со структурой причинно-следственных связей между явлениями в имеющемся у него сценарии.

Известно два основных подхода в моделировании: функциональный и алгоритмический [1–4]. При функциональном подходе главное, чтобы реакция модели на воздействие была подобна реакции объекта с заданной степенью точности. Соответствие структуры модели и объекта не обязательно. При алгоритмическом подходе, модель – это некоторое формальное представление сценария моделируемого процесса, которое имеется у предметного специалиста, и ее структура подобна структуре причинно-следственных связей между явлениями, описанными в сценарии. Основным средством достижения адекватности при функциональном подходе является подбор модели и ее идентификация путем нахождения требуемых параметров. Основным средством достижения адекватности для алгоритмической модели является коррекция ее структуры на основе уточнения исходного сценария.

Очевидно, что алгоритмический подход, с учетом описанных выше закономерностей, по самой своей сути обеспечивает большую степень доверия у пользователей, и алгоритмические модели будут ими легче восприниматься, что и наблюдается на практике.

Алгоритмический подход и алгоритмическое моделирование интенсивно развивались, особенно на начальном этапе их становления, в Санкт-Петербургском институте информатики и автоматизации РАН [4–6]. В рамках данного направления был разработан язык представления структуры алгоритмических моделей на основе “алгоритмических сетей” [5–6].

Если в логических функциональных схемах допустить в вершинах операторы любой природы, а не только логические, и обмениваться операторы будут не только логическими переменными, а переменными любого типа, то получится конструкция, которая называется алгоритмической сетью. Наличие элементов памяти (задержка) позволяет алгоритмическим сетям описывать динамические процессы, синтаксис соответствует синтаксису логических схем (на один вход оператора нельзя подавать одновременно два разных сигнала, недопустимы контуры без элементов памяти). Алгоритмическую сеть можно считать графическим представлением некоторой системы рекуррентных выражений, при этом типы переменных в ней могут быть произвольными, а не только числовыми. С другой стороны, алгоритмическая сеть отображает схему вычислений по данной системе рекуррентных выражений, позволяющую получить значения всех переменных, вычисляемых в ее вершинах.

Алгоритмические сети и алгоритмическое моделирование уже давно и успешно используются в СПИИРАН для автоматизации моделирования и для разработки моделей в различных предметных областях (экономика, экология, энергетика, кораблестроение, химические технологии и т.п.). На основе этого формализма было разработано несколько версий системы автоматизации моделирования КОГНИТРОН. Разработана алгебра алгоритмических сетей и алгоритмических моделей на их основе, в том числе и для распределенного моделирования (распределенные алгоритмические сети).

Алгоритмические сети унаследовали от функциональных логических схем важное свойство – они представляют процесс вычислений в максимально распараллеленном виде, поскольку вычисления в каждой вершине производятся сразу же, как становятся известны значения входных переменных. Как условие синхронизации процесса вычислений выступает требование одновременного срабатывания всех элементов задержки и переход к следующему такту вычислений только после окончания всех вы-

числений и во всех вершинах сети на данном такте вычислений (аналогия с тактовым импульсом в конечных автоматах). Значения входных переменных в течение одного такта не изменяются. Процедура планирования вычислений на алгоритмических сетях позволяет планировать вычисления для заданного числа используемых процессоров. Это позволяет использовать алгоритмические сети для автоматизации распараллеливания программ.

На структуру алгоритмических сетей наложены ограничения, для упрощения процесса синтеза программ, но даже с учетом этих ограничений класс алгоритмов, описываемых сетями, соответствует классу структурных (по Дейкстре) алгоритмов, то есть равномощен классу всех возможных алгоритмов. Рассмотрим алгоритмы, которые могут быть реализованы в алгоритмических сетях. Обычные алгоритмические сети, без ссылок в вершинах, соответствуют максимально распараллеленным по информационным связям линейным алгоритмам. Цикличность повторения обеспечивается вне алгоритмической сети (АС). То есть сеть описывает только тело некоторого цикла, условие останова которого задано в некотором программном шаблоне, в который помещается сгенерированное на основе сети тело цикла. Класс алгоритмов достаточно узок, но если разрешить ссылки в вершинах сети на другие сети, тогда, при необходимости, возможна реализация вложенных циклов, при этом каждый цикл будет привязан к своему програмному шаблону. Таким образом, реализовано две из конструкций Дейкстры. Конструкция ветвления получается также за счет ссылок на другие сети. В АС есть оператор, позволяющий присваивать своему выходу одно из двух возможных значений, в зависимости от выполнения некоторого отношения для третьей входной переменной. Если альтернативные значения будут именами других АС, на которые будет осуществляться ссылка, то это будет аналогом конструкции ветвления структурного программирования. Правда, переменные, получаемые на выходе, будут всегда иметь одинаковый идентификатор, но можно показать, что данный случай может быть сведен к эквивалентной классической конструкции.

Приведенные приемы несколько усложняют конструкцию АС, но зато позволяют избежать решения задач синтеза ветвящихся и циклических программ, которые были камнем преткновения для многих систем автоматизации програмирования, и осуществлять синтез только линейной части тел циклов.

Задана сеть с входами и выходами из неё. Известны размерности единиц измерения всех входов и выходов. Наша задача проверить, совпадут ли размерности входов и выходов после полного прохождения через сеть.

Существует ряд правил, которые должны соблюдатся в процессе проверки:

– если в вершине сумма или разность величин с размерностью $i$ , то на выходе будет величина с размерностью $i$;

– если в вершине произведение величин с размерностями $i$ и $j$ , то на выходе будет величина с размерностью $ij$;

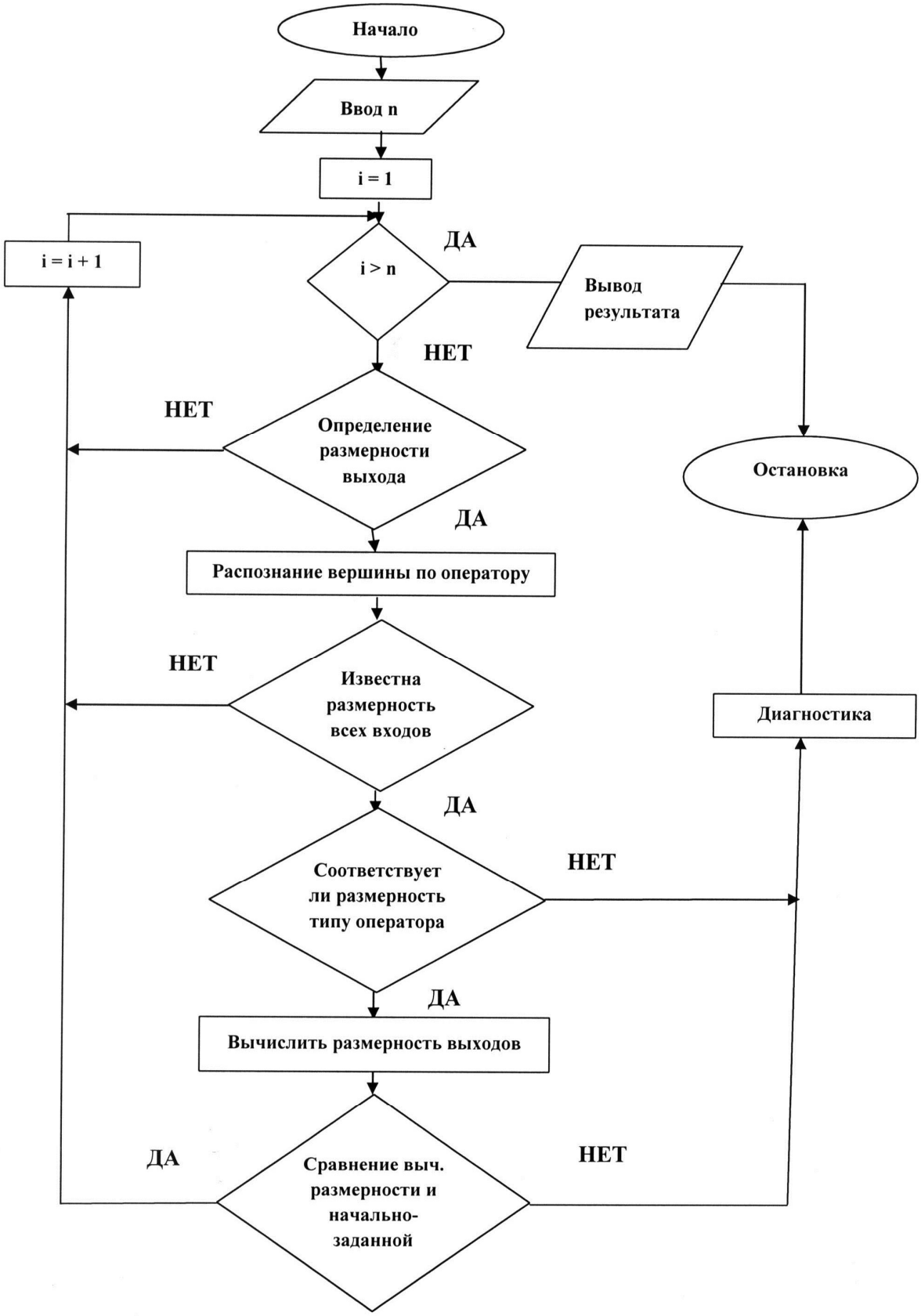
– если в вершине деление величин с размерностью $i$ и $j$, то на выходе у нас будет величина с размерностью $i/j$;

– при возведении в степень $n$ величины с размерностью $i$ в результате получим величину с размерностью $i^n$;

– если в вершине $\Delta t$, min или max , то на выходе размерность величины остаётся такой же;

– при логическом сравнении на выходе будет одна из размерностей величин, которую выбрали по определённому параметру сравнения.

Алгоритм проходит по всей сети, при этом проверяется на выполнение перечисленных правил, если они не выполняются, то остановка с диагностикой (рисунок).


Начало
Ввод n
i = 1
i = i + 1
i > n
ДА
Вывод результата
НЕТ
НЕТ
Определение размерности выхода
Остановка
ДА
Распознание вершины по оператору
НЕТ
Известна размерность всех входов
Диагностика
ДА
Соответствует ли размерность типу оператора
НЕТ
ДА
Вычислить размерность выходов
ДА
Сравнение выч. размерности и начально-заданной
НЕТ

Алгоритм «Цикл по вершинам сети»:

1. Цикл по вершинам (просматриваются все вершины, как таковые).
2. Определить размерность выхода.
3. Распознать вершину по оператору.
4. Если размерности всех входов известны, проверяем размерность величины на выходе. Иначе пропускаем вершину.
5. Соответствуют ли размерности типу оператора по соответствующим правилам. Иначе диагностика и остановка.
6. Произвести вычисления размерности выходов и занести вычисленную размерность как размерность данной величины.
7. Проверить соответствие вычисленной размерности и объявленной в начале размерности (если было объявлено). Если нет – диагностика и остановка.
8. Перейти к следующей вершине.

**Литература**

1. **Бирюков Б. В.** Кибернетика и методология науки. М.: Наука, 1974. 412 с.
2. **Поспелов Г. С., Ириков В. А., Курилов А. Е.** Процедуры и алгоритмы формирования комплексных программ. М.: Наука, 1985. 423 с.
3. Управление, информация, интеллект. / Под ред. А. И. Берга) М.: Мысль, 1976. 383 с.
4. Пономарев В.М. Алгоритмические модели в задачах исследования систем. // Алгоритмы и системы автоматизации исследований и проектирования. М.: Наука, 1980. С. 4–8.
5. **Иванищев В. В.** Алгоритмический базис для описания механизмов экономики. // Алгоритмические модели в автоматизации исследований и проектирования. М.: Наука, 1980. С. 37–42.
6. **Иванищев В. В., Марлей В. Е.** Основы теории алгоритмических сетей. СПб.: СПбГТУ, 2000. 180 с.