Вопрос №3.14

Автоматное программмирование. Отличие от абстрактных автоматов. Связь с машиной Тьюринга.

1 Автоматное программирование

1.1 Область применимости

Традиционные области применения автоматов в программировании:

- Копмиляторы, обработка строк — используются *абстрактные* автоматы.
- Системы управления — используются *структурные* автоматы.

Таким образом, даже традиционные области применения конечных автоматов охватывают принципиально различные классы программных систем. В действительности, круг задач, для которых целесообразно применять автоматный подходб гораздо шире. Наиболее точно он описывается через понятие *сложного поведения*.

Неформально можно сказать, что сущность (объект, подсистема) обладает сложным поведением, если в качестве реакции на некоторое входное воздействие она может осуществить одно из нескольких выходных воздействий. При этом существенно, что выбор конкретного выходного воздействия может зависеть не только от входного воздействия, но и от текущего состояния сущности и от предыстории. Для сущностей с *простым* поведением реакция на любое входное воздействие зависит только от этого воздействия.

В программных и программно-аппаратных вычислительных системах сущности со сложным поведением встречаются очень часто. Таким свойством обладают устройства управления, сетевые протоколы, диалоговые окна, персонажи компьютерных игр и многие другие объекты и системы. При традиционной реализации таких сущностей используются специальные логические переменные — *флаги*. Такой способ описания сложного поведения плохо структурирован, труден для понимания и модификации, подвержен ошибкам.

Одна из центральных идей автоматного программирования состоит в отделении описания *логики* сложного поведения (при каких условиях необходимо выполнить те или иные действия) от описания его *семантики* (собственно смысла каждого из действий). Кроме того, описание логики при автоматном подходе жестко структурировано (задается

конечным автоматом). Эти два свойства делают автоматное описание сложного поведения ясным и удобным.

Таким образом, автоматный подход целесообразно использовать для проектирования и реализации любой системы, в которой есть сложное поведением, однако только для тех компонент системы, которые являются сущностями со сложным поведением. Это стало возможно с появлением *объектно-ориентированого программирования с явным выделением состояний*. Этот подход не требует разрабатывать всю систему от начала и до конца в автоматном стиле, а позволяет органично интегрировать автоматные реализации сущностей со сложным поведением в существующую объектно-ориентированую систему.

1.2 Основные понятия

Базовым понятием автоматного программирования является *состояние*. Основное его свойство состоит в том, что текущее состояние несет в себе всю информацию о прошлом системы, необходимую для определения ее реакции на любое входное воздействие. При описании сложного поведения в терминах состояний знание предыстории для определения выходного воздействия уже не требуется.

В автоматном программировании различаются *управляющие* и *вычислительные* состояния сущности. Управляющих состояний обычно немного, они качественно отличаются друг от друга и определяют действие (алгоритм), которое совершает сущность в ответ на входное воздействие. Вычислительных состояний может быть бесконечно много, они отличаются лишь количественно и могут определять результат действия, но не алгоритм.

Понятие *входного воздействия* также является одним из базовых для автоматного программирования. Чаще всего, входное воздействие - это вектор. Его компоненты подразделяются на *события* и *входные переменные* в зависимости от смысла и механизма формирования. Совокупность конечного множества состояний и конечного множества входных воздействий образует (конечный) автомат без выходов. Такой автомат реагирует на входные воздействия, определенным образом изменяя текущее состояние. Правила, по которым происходит смена состояний, называют *функцией переходов* автомата.

То, что в автоматном программировании собственно и называется (конечным) автоматом, получается, если соединить понятие автомата без выходов с понятием *выходного воздействия*. Такой автомат реагирует на входное воздействие не только сменой состояния, но и формированием определенных значений на выходах. Правила формирования выходных

воздействий называют *функцией выходов* автомата (рис. 1).

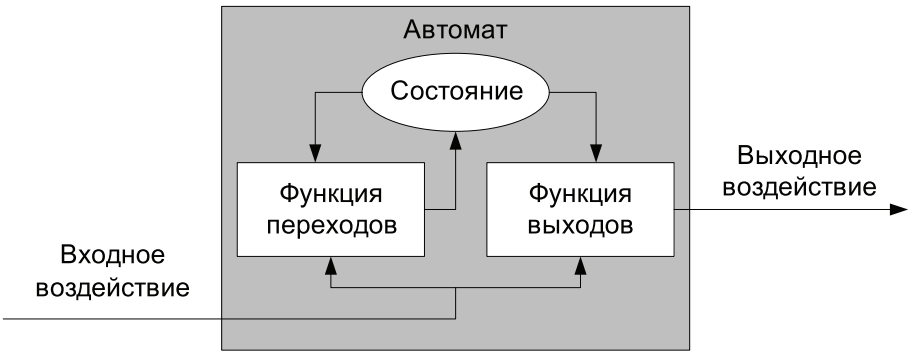

Рис. 1: Конечный автомат

При построении систем управления, которые обычно обладают сложным поведением, всегда выделяют управляющие устройства и управляемые объекты. Следуя этой концепции, любую сущность со сложным поведением естественно разделить на две части:

- управляющую часть, ответственную за логику поведения - выбор выполняемых действий, зависящий от текущего состояния и входных воздействий, а также за переход в новое состояние;

- управляемую часть, ответственную за выполнение действий, выбранных для выполнения управляющей частью, и, возможно, за формирование некоторых компонентов входных воздействий для управляющей части - обратных связей.

В соответствии с традицией теории управления, управляемая часть называется *объект управления*, а поскольку для реализации управляющей части используются автоматы, то она так и называется - *управляющий автомат* или просто *автомат*.

После разделения сущности со сложным поведением на объект управления и автомат реализовать ее уже несложно. Вся логика поведения сущности сосредоточена в управляющем автомате. Объект управления, в свою очередь, обладает простым поведением (а значит, может быть легко реализован традиционными «неавтоматными» методами).

Таким образом, в соответствии с автоматным подходом, сущности со сложным поведением следует представлять в виде *автоматизированных объектов управления* — так в теории управления называют объект управления, интегрированный вместе с системой управления в одно устройство.

Парадигма автоматного программирования состоит в представлении сущностей со сложным поведением в виде автоматизированных объектов управления.

Метод объектно-ориентированого проектирования с явным выделением состояний:

1. Производится объектная декомпозиция: система представляется в виде множества взаимодействующих сущностей. Для каждой сущности определяется ее интерфейс: набор событий, которые она обрабатывает (или, другими словами, сервисов, которые она предоставляет) и свойства этих сервисов с точки зрения клиентов.

2. Среди сущностей выбираются те, которые обладают сложным поведением. Для их описания будут использоваться автоматы, остальные сущности реализуются традиционным объектно-ориентированным методом.

3. Для каждой сущности со сложным поведением определяется небольшой набор *управляющих состояний*, соответствующих качественно различным режимам поведения сущности.

4. Определяются правила переходов между состояниями и осуществления выходных воздействий в терминах состояний, событий, а также запросов и команд объекта управления. Параллельно определяется интерфейс объекта управления как набор тех запросов и команд, которые используются при описании логики. Стоится структурная диаграмма (схема связей, диаграмма классов), отображающая взаимодействие управляющего автомата, объекта управления и внешней среды. Стоится граф переходов управляющего автомата.

5. Объект управления реализуется традиционным объектно-ориентированным методом.

6. Способ реализации управляющего автомата, вообще говоря, не является частью метода. Это может быть как использование любого из множества существующих шаблонов проектирования (от конструкций switch до шаблона State), так и автоматическая генерация кода по графу переходов (это предпочтительно).

2 Отличие от абстрактных автоматов. Связь с машиной Тьюринга

В теории формальных языков изучаются, так называемые, *абстрактные автоматы* (называемые также *абстрактными машинами* или *абстрактными вычислителями*). Интерес при их изучении представляет только вычислительная мощность — класс языков, которые может распознавать машина.

В логическом управлении рассматриваются *структурные автоматы*. Они выступают в роли управляющих устройств в системах управления. Интерес здесь представляет число параллельных входов и выходов автомата, его связи с объектом управления и другими элементами системы, простота реализации.

При использовании автоматов в программировании не имеют значения ни исследования вычислительной мощности, ни минимизация числа функциональных элементов и вопросы кодирования. В этой области нас интересует только то, как использовать конечные автоматы для построения корректных и надежных программ. Автоматное программирование не просто утверждает, что для реализации некоторых конструкций в программировании могут применяться конечные автоматы, но предлагает универсальный метод проектирования и разработки, существенно упрощающий описание сложного поведения.

Примеры абстрактных автоматов: детерминированный и недетерминированный конечный автомат, автомат со спонтанными переходами, автомат с магазинной памятью, машина Тьюринга (и ее модификации), счетчиковые машины и т.д.

Все они состоят из устройства управления (представляющего собой ДКА с выходом) и хранилища данных того или иного вида (ленты, магазина). Для теории формальных языков вид хранилища и набор элементарных операций с данными имеют решающее значение: они определяют вычислительную мощность машины. При моделировании и высокоуровневой программной реализации сущностей со сложным поведением, удобнее заменить конкретное хранилище данных объектом управления (ОУ), множество (вычислительных) состояний которого может быть любым и определяется спецификой решаемой задачи. Вместо ограниченного набора элементарных операций с данными удобно использовать произвольные запросы и команды (рис. 2).

Достоинство машины Тьюринга в отделении конечного управляющего устройства от бесконечной дополнительной памяти — это мы сохраним и в автоматном программировании. Однако машина Тьюринга непри-

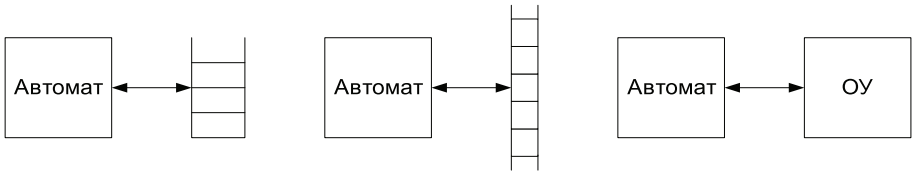

Рис. 2: Слева направо: автомат с магазинной памятью, автомат с ленточной памятью (машина Тьюринга), автомат с произвольной памятью (автоматизированный объект)

годна для описания сущностей со сложным поведением в реальных программных системах, так как набор элементарных операций с памятью у нее слишком прост. В результате на ней слишком сложно программировать, поскольку автомату приходится не только управлять, но и выполнять не свойственную ему функцию вычисления. Переход к модели автоматизированного объекта управления позволяет использовать в качестве элементарных операций произвольные запросы и команды, при этом на управляющий автомат ложится лишь часть ответственности по реализации алгоритма: та, что связана с логикой (управлением). Выбор уровня абстракции элементарных операций при моделировании сущности со сложным поведением определяет разделение поведения на логику и семантику, а состояний на управляющие и вычислительные. Это и есть наиболее творческий и нетривиальный шаг в автоматном подходе к разработке программного обеспечения.

Определение. *Автоматизированным объектом управления* (рис. 3) называется пара $< A, O >$, где A — управляющий автомат, O — объект управления.

Управляющий автомат представляет собой шестерку

$$< X, Y, Z, y_0, \varphi, \delta >,$$

где $X = X_E \times X_O$ — конечное множество входных воздействий, причем каждое входное воздействие x состоит из компоненты x_E, порождаемой внешней средой, и компоненты x_O, порождаемой объектом управления; Y — конечное множество управляющих состояний; Z — конечное множество выходных воздействий; $y_0 \in Y$ — начальное состояние; $\varphi = \varphi' \times \varphi''$ — функция выходов (выходных воздействий), состоящая, в общем случае, из двух компонент: функции выходных воздействий в состояниях $\varphi' \colon Y \to Z$ и функции выходных воздействий на переходах $\varphi'' \colon X \times Y \to Z$; $\delta \colon X \times Y \to Y$ - функция переходов.

Объект управления — это тройка

$$< V, f_q, f_c >,$$

где V — потенциально бесконечное множество вычислительных состояний (или значений), f_q — функция, сопоставляющая входное воздействие вычислительному состоянию, f_c —функция, изменяющая вычислительное состояние в зависимости от выходного воздействия.

Функции f_q и f_c являются математическими эквивалентами набора запросов и набора команд соответственно.

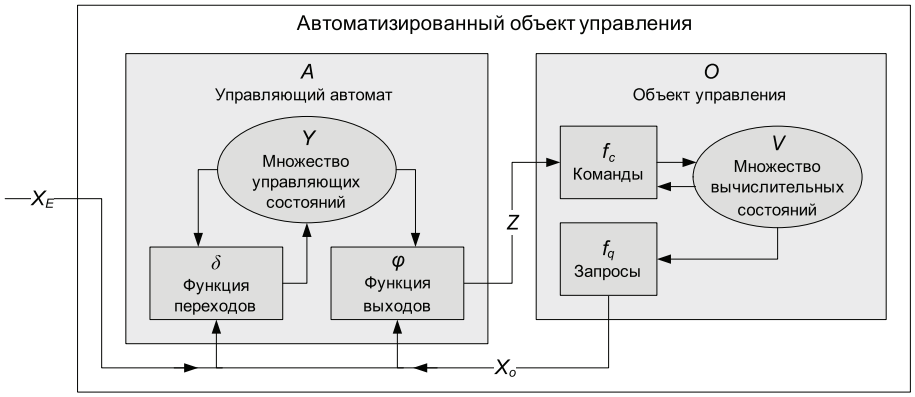

Рис. 3: Автоматизированный объект управления