

Википедия

Switch-технология

Материал из Википедии — свободной энциклопедии

Switch-технология — технология разработки систем логического управления на базе конечных автоматов, охватывающая процесс спецификации, проектирования, реализации, отладки, верификации^[1], документирования и сопровождения. Предложена А. А. Шалыто в 1991 году^[2].

Содержание

Подход к алгоритмизации

Используемые понятия

- Граф переходов
- Схема связей
- Кодирование состояний
- Структура программы
- Взаимодействия автоматов

Универсальность

Стили программирования

Смежные технологии

- Нейронные сети и генетические алгоритмы
- Параллельные вычисления

Проверка, отладка и верификация автоматных программ

- Проверка и отладка
- Верификация

Реализация и инструментальные средства

- Реализация графов переходов
- Инструментальные средства

Применение технологии

Проект «Технология автоматного программирования: применение и инструментальные средства»

См. также

Литература

Примечания

Подход к алгоритмизации

В качестве языков алгоритмизации и программирования в системах логического управления в зависимости от типов управляющих вычислительных устройств применяются: алгоритмические языки высокого уровня (C, Pascal, Forth, …), алгоритмические языки низкого уровня (Ассемблеры), и специализированные языки (например, на базе лестничных и функциональных схем).

Кроме того, существуют два подхода алгоритмизации систем этого класса. В первом из них считается, что известен алгоритм функционирования объекта управления и требуется по нему синтезировать алгоритм логического управления, обеспечивающий заданное поведение: «Для того чтобы клапан открылся, вычислитель должен на вход открытия исполнительного механизма клапана подавать единичный сигнал». Во

втором подходе учитывается информация о состоянии объекта управления: «Если вычислитель подает на вход открытия исполнительного механизма клапана единичный сигнал, то клапан открывается». Таким образом, первая стратегия базируется на понятии «состояние», а вторая — на понятии «событие».

Ни та, ни другая разновидность управления в общем случае не является исчерпывающей, однако в Switch-технологии предлагается сделать понятие «состояние» первичным, а в качестве языка спецификации для описания алгоритмов применять графы переходов — предлагается представлять программу как систему взаимодействующих конечных автоматов, описываемых графами переходов. Другими словами, первоначально описывается поведение управляющего устройства, а не его структура, а построение алгоритмов и программ начинается с формирования состояний.

Объекты управления, как и система управления, могут быть реализованы с помощью рассматриваемой технологии, что позволяет моделировать систему в целом.

Используемые понятия

Граф переходов

Поведение автоматов задается графами переходов (диаграммами состояний), на которых для их компактности входные и выходные воздействия обозначаются символами, а слова используются только для названий пронумерованных состояний. Расшифровка символов выполняется на схеме связей. Применение символов позволяет изображать сложные графы переходов весьма компактно — так, что человек может в большинстве случаев охватить каждый из них одним взглядом. Это обеспечивает когнитивное восприятие указанных графов.

Графы переходов в наглядной для человека форме отражают переходы между состояниями, а также «привязку» выходных воздействий и других автоматов к состояниям и/или переходам. Для упрощения изображения графов переходов допустимо использование составных состояний, которые применяются в тех случаях, когда несколько вершин имеют одинаково помеченные исходящие дуги.

Задание поведения программ с помощью графов переходов позволяет проверять корректность их построения, например, полноту, непротиворечивость и отсутствие генерирующих контуров.

В программе, построенной по графам переходов, также, как и в этих графах, используются символьные обозначения, а не смысловые идентификаторы, как в других стилях программирования. Это связано с тем, что текст программы в рамках указанной технологии строится по графу переходов формально и изоморфно, а изменения вносятся не непосредственно в текст программы, а только после корректировки схемы связей и графа переходов. Изложенное позволяет обеспечить синхронность изменений программ и их проектной документации.

Схема связей

В качестве основного документа, определяющего структуру программы, как и при автоматизации технологических (и не только) процессов, в Switch-технологии используется схема связей^[3]. Она может совмещаться со схемой взаимодействия автоматов.

Схема связей определяет интерфейс автоматов и позволяет применять в графах переходов и в реализующих их программах символьные обозначения. Для объектно-ориентированных программ с явным выделением состояний диаграммы классов в рамках рассматриваемого подхода изображаются в виде схем связей.

Кодирование состояний

Состояния кодируются для того, чтобы различать их. Это особенно важно, когда в разных состояниях формируются одинаковые значения выходных переменных. Могут использоваться различные виды кодирования состояний, но в качестве основного используется многозначное кодирование, позволяющее кодировать состояния любого автомата с помощью только одной переменной^[4]. При этом её значность должна быть равна числу состояний рассматриваемого автомата, а каждому состоянию присваивается соответствующий номер. Кодирование состояний позволяет отказаться от флагов, которые неявно выполняют ту же функцию.

Возможность слежения за состояниями автомата по значениям одной многозначной переменной позволяет ввести в программирование (по аналогии с теорией управления) понятие «наблюдаемость»^[5].

Структура программы

При использовании Switch-технологии программы (их схемы) должны строиться, начиная с дешифратора состояний, а не с дешифратора входных воздействий, как это делается при использовании других стилей программирования^[6].

Существуют три схемы реализации автоматов:

- Четырехуровневая схема программ (дешифратор состояний — выходные воздействия — дешифратор входных воздействий — формирование следующих состояний) реализует автоматы Мура.
- Четырехуровневая схема программ (дешифратор состояний — дешифратор входных воздействий — выходные воздействия — формирование следующих состояний) реализует автоматы Мили.
- Пятиуровневая схема программ (дешифратор состояний — выходные воздействия — дешифратор входных воздействий — выходные воздействия — формирование следующих состояний) реализует смешанные автоматы.

Схемы программ с указанной структурой названы автоматными. При таком построении граф переходов, автоматная схема программ и конструкция языка программирования аналогичная конструкции switch языка С, реализующая дешифратор состояний, изоморфны.

Взаимодействия автоматов

Автоматы могут взаимодействовать по вложенности (один автомат вложен в одно или несколько состояний другого автомата), по вызываемости (один автомат вызывается с определенным событием из выходного воздействия, формируемого при переходе другого автомата), по обмену сообщениями (один автомат получает сообщения от другого) и по номерам состояний (один автомат проверяет, в каком состоянии находится другой автомат). Вложенность может рассматриваться как вызываемость с любым событием. Ни число автоматов, вложенных в состояние, ни глубина вложенности не ограничены.

Взаимодействие автоматов отражается на «схеме взаимодействия автоматов»^[3], которая может совмещаться со «схемой связей». Проверка взаимодействия автоматов может выполняться протоколированием их работы.

Универсальность

Как известно каждый алгоритм можно реализовать на машине Тьюринга. Рассматриваемая технология является расширением этой модели для практического использования. В случае машины Тьюринга входное воздействие с ленты подается на конечный автомат, а на ленту записывается выходное воздействие. Сложность программирования на машине Тьюринга в основном определяется тем, что в ней объект управления (ячейка) очень прост, и в ней автомату приходится выполнять не только присущую ему функцию — управление, но, при необходимости, и обеспечивать вычисления. Если заменить ленту любым более сложным устройством, а автомат — системой взаимосвязанных автоматов, то получится обобщение

машины Тьюринга, которое может быть использовано для практического программирования. Таким образом, предлагаемый подход является универсальным средством для реализации алгоритмов.

Стили программирования

Стили программирования различаются по базовым понятиям^[7], в качестве которых используются такие понятия как «событие», «подпрограмма», «функция», «класс» («объект») и т. д. В Switch-технологии таким понятием является «состояние». При этом события рассматриваются лишь как средства для изменения состояний. Стил программирования, основанный на явном выделении состояний и применении автоматов для описания поведения программ, назван «автоматное программирование»^[2], а соответствующий стиль проектирования программ — «автоматное проектирование программ». Автоматное программирование можно рассматривать не только как самостоятельный стиль программирования, но и как дополнение к другим стилям, например, к объектно-ориентированному.

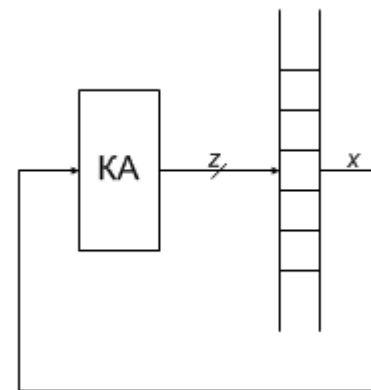


Рис. 1. Машина Тьюринга, как Конечный автомат с входными и выходными воздействиями, получаемыми и записываемыми с и на ленту соответственно

Смежные технологии

Нейронные сети и генетические алгоритмы

Области применения автоматов и нейронных сетей обычно различны. Однако существуют задачи, в которых целесообразно их совместное применение^[8].

Известны также задачи, которые могут решаться с помощью любого из указанных средств. В этом случае применение автоматов более целесообразно, так как в них легко вручную вносить изменения. Это может быть важно, например, в случае, когда закон управления строится в два этапа, первый из которых состоит в генерации автоматов с помощью генетических алгоритмов, а второй — в корректировке автоматов в результате экспериментов.

Параллельные вычисления

В программах, построенных традиционным путём, выделение фрагментов, которые могут быть реализованы параллельно, является весьма трудной задачей. Существует широкий класс задач, спецификация которых осуществляется с помощью параллельно работающих автоматов, которые обмениваются сообщениями. Такие автоматы эффективно реализовывать на основе систем с параллельными вычислениями^[9].

Проверка, отладка и верификация автоматных программ

Проверка и отладка

Программа, изоморфная графу переходов, может быть проверена сверкой её текста с этим графом. Сертификация программы логического управления, которая реализована указанным образом по графу переходов автомата, являющегося автоматом Мура, может выполняться в два этапа: проверка в динамике по значениям многозначной внутренней переменной, кодирующей состояние автомата, наличия всех переходов

в графе; проверка в статике соответствия значений выходных переменных с их значениями, указанными в вершинах графа переходов^[4]. Такая проверка более корректна, чем традиционная проверка «вход-выход». Отладка автоматных программ выполняется в автоматных терминах, что резко её упрощает.

Верификация

Программы, создаваемые на основе Switch-технологии, могут быть эффективно верифицированы методом Model checking^[10], так как в таких программах управляющие состояния явно выделены, а их количество обозримо. Это позволяет строить компактные модели Крипке даже для программ большой размерности.

Структура автоматных программ, в которых функции входных и выходных воздействий почти полностью отделены от логики программ, делает практичным верификацию этих функций на основе формальных доказательств с использованием пред- и постусловий^{[11][12]}.

Реализация и инструментальные средства

Реализация графов переходов

Графы переходов реализуются формально и изоморфно, например, с помощью конструкции switch (например, языка Си) по соответствующим шаблонам (поэтому предлагаемый подход был назван «Switch-технология»).

В конструкции switch используются символы входных и выходных воздействий, а не функции их реализующие, которые размещаются отдельно. Такая декомпозиция упрощает понимание логики программы за счет компактного её представления и соответствует правилу «отделения логики переключения от деталей того, что происходит»^[13].

Для каждого автомата создается четыре документа: словесное описание алгоритма (декларация о намерениях), схема связей, граф переходов, изоморфная реализация.

Реализация выполняется так, что в каждом программном цикле или при каждом запуске автомата в нем выполняется не более одного перехода. Это делает номер каждого состояния автомата доступным для его «окружения» и позволяет не вводить новых переменных для обеспечения взаимодействия автоматов^[4].

Инструментальные средства

Возможность быстрого, безошибочного, автоматического и изоморфного перехода от набора графов переходов к программе на языке высокого уровня позволяет значительно упростить отладку и моделирование управляющих автоматов и делает программу полностью наблюдаемой и управляемой.

Известно большое число инструментальных средств для генерации программ, реализующих графы переходов (en:List of state machine CAD tools). Приведем список наиболее известных проектов:

- Visio2Switch — инструментальное средство Visio2Switch (<http://is.ifmo.ru/progeny/visio2switch/>) позволяет по графу переходов, построенному в определенной нотации и изображенному с помощью редактора Visio, автоматически реализовать его в виде изоморфной программы на языке C.
- MetaAuto — инструментальное средство MetaAuto (<http://is.ifmo.ru/projects/metaauto/>) позволяет по графу переходов, построенному в той же нотации и изображенному с помощью того же редактора, автоматически реализовать его в виде изоморфной программы на любом языке программирования, для которого предварительно построен шаблон.
- UniMod — инструментальное средство UniMod (<http://unimod.sourceforge.net/>) предназначено для поддержки автоматного программирования и построения и реализации не только автоматов, но и программ в целом.

Это средство характеризуется формулой: «UniMod = UML + Switch-технология + Eclipse + Java + SourceForge». Оно является открытым и использует только два типа UML-диаграмм: диаграммы классов в форме схем связей (для описания структуры программы) и диаграммы состояний (для описания её поведения)^{[14][15]}. Валидация диаграмм состояний выполняется автоматически. Указанное инструментальное средство позволяет создавать программы, поддерживая концепции «Исполняемый UML» (en:Executable UML) и «Разработка на базе моделей» (en:Model Driven Architecture).

При использовании инструментального средства UniMod программа состоит из указанных диаграмм и написанных вручную функций, соответствующих входным и выходным воздействиям. Таким образом, это инструментальное средство позволяет совмещать декларативный и процедурный подходы к написанию программ.

Реализация таких программ может быть выполнена как в режиме интерпретации, так и в режиме компиляции. Это средство реализует концепцию «Визуальное конструирование программ»^[16].

Применение технологии

Описанный подход используется в четырех направлениях:

- логическое управление (события отсутствуют, входные и выходные переменные двоичны);
- программирование с явным выделением состояний;
- объектно-ориентированное программирование с явным выделением состояний;
- вычислительные алгоритмы (алгоритмы дискретной математики).

Известные практические применения:

- Впервые технология автоматного программирования применительно к логическому управлению изложена в книге^[4] применительно к логическому управлению.
- Программирование с явным выделением состояний для систем, реагирующих на события, впервые было использовано при автоматизации судовых дизель-генераторов^[17].
- Объектно-ориентированное программирование с явным выделением состояний впервые было применено в игре Robocode^[18].
- Разработка прикладного программного обеспечения для микроконтроллеров^[19]
- Проекты, созданные с использованием инструментального средства UniMod^[20].

Кроме того, автоматный подход эффективен для реализации визуализаторов алгоритмов дискретной математики и при реализации некоторых из таких алгоритмов (например, обход дерева^[21]), а также все чаще начинает использоваться в рамках такого научного направления, как «искусственный интеллект» [35], и при программировании мобильных устройств^[22].

Проект «Технология автоматного программирования: применение и инструментальные средства»

В 2005 году по результатам конкурса, проводимого в рамках Федеральной целевой научно-технической программы «Исследования и разработки по приоритетным направлениям развития науки и техники» на 2002—2006 годы, проект «Технология автоматного программирования: применение и инструментальные средства» был поддержан Федеральным агентством по науке и инновациям.

Проект вошел в список 15 наиболее перспективных и социально значимых проектов, выполняемых в рамках указанной программы (проект ИТ-13.4/004^[23], а также статья в приложении к газете Коммерсантъ^[24]).

Указанные выше работы в области Switch-технологии находятся в русле работ по обеспечению высокого качества программного обеспечения, проводимых в Западной Европе при создании синхронного программирования для ответственных систем^[25] и в NASA при создании программного обеспечения для беспилотных космических аппаратов^[26].

См. также

- Конечный автомат
- Автоматное программирование

Для поддержки автоматного программирования создан сайт <http://is.ifmo.ru/>.

Литература

- Шалыто А.А.* Switch-технология. Алгоритмизация и программирование задач логического управления. (<http://project.ifmo.ru/books/1>). — СПб.: Наука, 1998. — 628 с.

Примечания

↑ Показывать компактно

- Лукин М. А. // ВЕРИФИКАЦИЯ ПАРАЛЛЕЛЬНЫХ АВТОМАТНЫХ ПРОГРАММ (<http://ntv.ifmo.ru/file/article/8323.pdf>) . - Статья. - Научно-технический вестник ИТМО. - УДК 004.05
- Шалыто А. А. Программная реализация управляющих автоматов //Судостроительная промышленность. Серия «Автоматика и телемеханика». 1991. Вып.13, с.41,42
- Туккель Н. И., Шалыто А. А. SWITCH-технология – автоматный подход к созданию программного обеспечения «реактивных» систем //Программирование. 2001. № 5, с.45-62.
- Шалыто А. А. Switch-технология. Алгоритмизация и программирование задач логического управления. СПб.: Наука. 1998.
- Шалыто А. А. Алгоритмизация и программирование для систем логического управления и «реактивных» систем //Автоматика и телемеханика, 2001, № 1, с.3–39.
- Шалыто А. А. Использование граф-схем и графов переходов при программной реализации алгоритмов логического управления. II //Автоматика и телемеханика. 1996. № 7. с.144-169.
- Непейвода Н. Н. Стили и методы программирования. М.: Интернет-Университет Информационных технологий. 2005
- Кретинин А. В., Солдатов Д. В., Шалыто А. А., Шостак А. В. Ракеты. Автоматы. Нейронные сети // Нейрокомпьютеры: разработка и применение. 2005. № 5, с. 50-59.
- FSM (http://itc.ua/articles/fsm_19921/). ITC Publishing (20 февраля 2005). Проверено 8 апреля 2012. Архивировано (https://www.webcitation.org/687RMY65V?url=http://itc.ua/articles/fsm_19921/) 2 июня 2012 года.
- Кларк Э., Грамберт О., Пелед Д. Верификация моделей программ: Model Checking. М.: МЦНМО. 2002.
- Дейкстра Э. Заметки по структурному программированию / Дал У., Дейкстра Э., Хоор К. Структурное программирование. М.: Мир, 1975.
- Мейер Б. Объектно-ориентированное конструирование программных систем. М.: Русская редакция. 2005.
- Фаулер М. Рефакторинг. Улучшение существующего кода. СПб.: Символ. 2003.
- Гуров В. С., Мазин М. А., Нарвский А. С., Шалыто А. А. UML. SWITCH-технология. Eclipse // Информационно-управляющие системы. 2004. № 6, с.12-17.
- Gurov V.S., Mazin M.A. , Narvsky A.S., Shalyto A.A. UniMod: Method and Tool for Development of Reactive Object-Oriented Programs with Explicit States Emphasis /Proceedings of St. Petersburg IEEE Chapters. Year 2005. International Conference «110 Anniversary of Radio Invention». SPb ETU «LETI». 2005. V. 2, pp. 106—110.
- Новиков Ф. А. Визуальное конструирование программ //Информационно-управляющие системы. 2005. № 6. с.9–22.
- Туккель Н. И., Шалыто А. А. Система управления дизель-генератором (фрагмент). Программирование с явным выделением состояний. Проектная документация. 2002. с. 51
- Туккель Н. И., Шалыто А. А. Система управления танком для игры «Robocode». Вариант 1. Объектно-ориентированное программирование с явным выделением состояний. 2001. с. 52
- Татарчевский В. (Екатеринбург) Применение SWITCH-технологии при разработке прикладного программного обеспечения для микроконтроллеров. Части 1-7 //Компоненты и технологии. 2006. № 11 — 2007. № 7.

20. UniMod-проекты. <http://is.ifmo.ru/unimod-projects/>
21. Корнеев Г. А., Шамгунов Н. Н., Шалыто А. А. Обход деревьев на основе автоматного подхода // Компьютерные инструменты в образовании. 2004. № 3, с.32-37.
22. Программирование мобильных устройств. <http://is.ifmo.ru/science/MD-Mobile.pdf>
23. Выбраны лучшие инновационные проекты России //Информационная система «Наука и Инновации», <http://www.rsci.ru/company/innov/more.html?MessageID=965>
24. Волшебный сундучок Роснауки //Приложение к газете Коммерсантъ, 16.11.2005.
<https://web.archive.org/web/20070609080612/http://www.kommersant.ru/application.html?DocID=625381>
25. The Synchronous Languages 12 Years Later. [Архивированная копия \(http://www-sop.inria.fr/aoste/benveniste2003synchronous.pdf\)](http://www-sop.inria.fr/aoste/benveniste2003synchronous.pdf) (недоступная ссылка — *история* (https://web.archive.org/web/*/http://www-sop.inria.fr/aoste/benveniste2003synchronous.pdf)). Проверено 23 июня 2006. Архивировано (<https://web.archive.org/web/20060715224034/http://www-sop.inria.fr/aoste/benveniste2003synchronous.pdf>) 15 июля 2006 года. (англ.)
26. Риган П., Хемилтон С. NASA: миссия надежна //Открытые системы, 2004. № 3.
<http://www.osp.ru/os/2004/03/184060/>

Источник — <https://ru.wikipedia.org/w/index.php?title=Switch-технология&oldid=92702299>

Эта страница в последний раз была отредактирована 17 мая 2018 в 05:59.

Текст доступен по [лицензии Creative Commons Attribution-ShareAlike](#); в отдельных случаях могут действовать дополнительные условия.

Wikipedia® — зарегистрированный товарный знак некоммерческой организации [Wikimedia Foundation, Inc.](#)