РЕАГИРУЮЩИЕ СИСТЕМЫ (*R*-СХЕМЫ)

Использования для моделирования информационно-вычислительных систем классических диаграмм конечных автоматов Мили-Мура требуют создания для каждого состояния отдельных узлов, что может приводить к очень большому числу узлов и переходов между узлами при моделировании чуть ли не самых простых систем. Такая тенденция к сложности снижает удобочитаемость диаграммы состояний.

Ещё одно из ограничений моделирования информационно-вычислительной системы как обычного конечного автомата – это отсутствие в диаграммах спецификации действий, вызванных переходом из одного состояния в другое. Кроме того, моделирование традиционных конечных автоматов основано на концепции последовательных переходах из одного состояния к следующему и наличие параллельных структур не допускается. Это серьёзно ограничивает их применение для моделирования поведения сложных информационно-вычислительных систем, так как на разных уровнях рассмотрения сложная система может одновременно находиться в различных параллельных состояниях.

Ограниченность применения обычных конечных автоматов для моделирования поведения сложных информационно-вычислительных систем и одновременно свойственная конечным автоматам наглядность описания их поведения с помощью диаграмм состояний и переходов послужило тому, что в 1991 году Грейди Бучем на основе работ Дэвида Харела (D. Harel)[1] был разработан класс моделей так называемых *реагирующих систем* (reactive system) – *R*-схемы, обобщающие поведение автоматных моделей различных классов: *F*-схем, *P*-схем и *Q*-схем, так что это позволило рассматривать *R*-схемы как язык диаграмм для адекватного описания поведения сложных информационно-вычислительных систем.

Реагирующая система (reactive system) — это система, постоянно

1 Дэвид Харел, Институт Вейцмана, информатика и прикладная математика, преподаватель. Создатель системы обозначений Харела (Harel. Statecharts. 1987, с. 231-274; Harel. On Visual Formalism. 1988, с. 514-530).

ожидающая внешних или внутренних событий и реагирующая на них. Реагирующие системы являются типичными системами дискретно-событийного типа: события происходят в дискретные моменты времени, реакция системы на события состоит в изменении переменных состояния этих систем и формально такая реакция является мгновенной.

Поведение реагирующей системы может рассматриваться в любой из моделей времени: в непрерывном времени - $M_{\mathcal{T}}(t_0,\delta t)$, в реального времени - $M_{\mathcal{T}}(t_0,\Delta t)$, или виртуальном времени - $M_{\mathcal{T}}(0,\not\geq)$, и интерпретируется как изменение её состояния в ответ на возникающие контролируемые события, и выполнение различных действий при переходе из текущего состояния в новое состояние.

Моделирование реагирующих систем осуществляется на графическом языке *диаграмм состояний*, которые ещё называют стейтчарты (*statechart*) Харела. Диаграммы состояний реагирующих систем позволяют предельно полно выразить логику поведения предмета моделирования, как событийное управление выполнением абстрактных действий. В частности диаграммы состояний реагирующих систем применяются для моделирования программ с целью их верификации и даже используются как графический язык автоматного программирования[2].

Диаграммы состояний и переходов

В диаграммах состояний *состояния* представляются в виде прямоугольников или овалов, а *переходы* - ориентированными дугами. Система может находиться в каждый момент времени только в одном состоянии. Переходы из состояния в состояние случаются, если происходит некоторое контролируемое *событие*, инициирующее этот переход, и выполняется *условие* (предикат), разрешающее переход в текущий момент времени. Если условие перехода в текущий момент времени не выполняется, то система сохраняет своё текущее состояние.

[2] **Автома́тное программи́рование** — парадигма программирования, при использовании которой программа или её фрагмент осмысливается как модель какого-либо формального автомата.

В реагирующей системе множество событий определяется как для автоматной модели. Событием перехода может быть, например, истечение таймаута, переключение в истину некоторого заданного условия (предиката), определённого на переменных модели и т.п. С переходом может быть связано некоторое *действие перехода* - изменение переменных, посылка сигнала и т. п., которое выполняется в результате реализации этого перехода. С каждым состоянием также могут быть связаны *действия*. Одно действие выполняется в момент *входа* в это состояние, другое действие выполняется в момент *выхода* из состояния в результате срабатывания перехода.

На диаграмме (рисунок 1) система из состояния A переходит в состояние B, если возникает контролируемое событие a и при этом выполняется условие - предикат P="Истина".

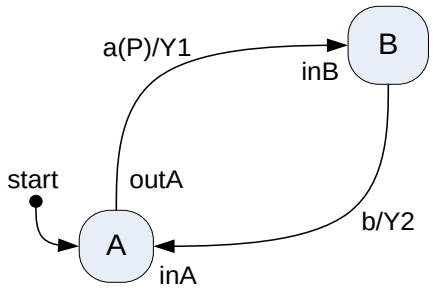

Рисунок 1 Простая диаграмма переходов

Короткая стрелка-указатель с идентификатором start, входящая в состояние A, говорит о том, что это состояние начальное: в начальный момент времени система будет находиться именно в этом состоянии. Очевидно, что у системы может быть ровно одно начальное состояние. На диаграмме указаны также действия, выполняемые при срабатывании соответствующих переходов: a(P), b; которые на диаграмме обозначены Y1 и Y2. Кроме того, для состояния A определены действия, связанные с входом в состояние A – inA, или выходом из состояния A – outA. Определено действие inB, связанное с входом в состояние B. Действие, связанное с выходом из состояния B не определено (являются пустыми).

Язык диаграмм состояний реагирующих систем позволяет моделировать и сложные системы на различных уровнях их представления. Если реагирующая

система рассматривается как сложная система, то при её моделировании кроме *простых* состояний, не имеющих внутренней структуры, используются *сложные* состояния с инкапсулированной внутренней структурой, в свою очередь выражающейся диаграммой состояний. Различают три типа сложных состояний с внутренней структурой:

- иерархические состояния (гиперсостояния),
- исторические состояния,
- состояния условных переходов.

Иерархические состояния

Иерархические или гиперсостояния вводятся для того, чтобы объединить несколько состояний реагирующей системы, имеющих одинаковые реакции на некоторые события, в отдельную подсистему, интерпретируемую как *гиперсостояние*. Каждое гиперсостояние требует, чтобы одно из включённых в него состояний было помечено как единственное начальное. Гиперсостояния могут включать в себя как простые (элементарные), так и вновь созданные гиперсостояния. В итоге в общем случае структура гиперсостояния может представлять собой многоуровневую иерархическую систему вложенных друг в друга состояний различных уровней.

Ниже приведён пример использования гиперсостояния (рисунок 2).

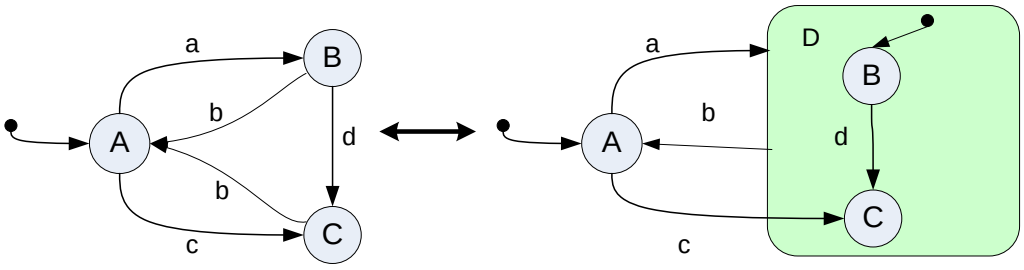

Рисунок 2 Пример использования гиперсостояния

Слева на рисунке представлен вариант диаграммы, в котором из состояний B и C по событию b осуществляется безусловный переход в состояние A. Справа представлен эквивалентный этой диаграмме диаграмма состояний с гиперсостоянием D, объединившим в себе элементарные состояния B и C,

связанные переходом d. Теперь из состояния A реагирующая система по событию a переходит в гиперсостояние D, а по событию c – в простое состояние C внутри гиперсостояния D. Переход из A в D по событию a эквивалентен переходу из A в B как в начальное состояние гиперсостояния D. Заметим, что теперь имеется единственный переход по событию b из гиперсостояния D в состояние A, в каком бы при этом внутреннем состоянии гиперсостояния D система не находилась. То есть, переход из гиперсостояния D по событию b в состояние A происходит из любого текущего внутреннего состояния.

Такая эквивалентная модификация исходной диаграммы состояний, введя гиперсостояние D, позволила скрыть "лишние" детали и распределить фокус внимания на поведение системы по уровням иерархии. В итоге не только сокращается количество переходов, рассматриваемых на одном уровне иерархии, но и упрощается понимание многоцелевой работы моделируемой системы на различных семантических уровнях иерархии.

Заметим, что начальным состоянием реагирующей системы в обеих диаграммах остаётся состояние A, а состояние B является начальным только при входе в гиперсостояние D.

Исторические состояния

Так называемое *историческое состояние* включается в состав гиперсостояния, чтобы отразить в диаграмме безусловные асинхронные переходы (спонтанные) из гиперсостояния в другое состояние диаграммы с возвратом в исходное гиперсотояние в его прерванное внутреннее состояние. Возможность выражения такой логики поведения необходимо для моделирования программ, содержащих программные модули, работающие в режиме прерывания, например, драйверов устройств операционной системы. Необходимость в историческом состоянии возникает и при моделировании параллельного выполнения программных процессов в однопроцессорной вычислительной системе, когда процессор асинхронно переключается с выполнения одного программного процесса на выполнение другого по истечении выделенного текущему процессу кванта процессорного времени. После того, как вновь активизированный для

выполнения процесс исчерпает свой квант процессорного времени, процессор будет предоставлен ранее прерванному процессу для продолжения его выполнения с прерванного места. То есть, после асинхронного перехода системы из некоторого состояния внутри гиперсостояния, вызванного событием прерывания, следует событие возврата в то же самое место внутри гиперсостояния.

Историческое состояние предназначено для запоминания и хранения в гиперсостоянии ссылки на его внутреннее состояние, в котором гиперсостояние находилась, когда диаграмма покинула его, перейдя асинхронно в другое состояние. Ниже (рисунок 3) приведён пример гиперсостояния I с историческим состоянием H внутри него.

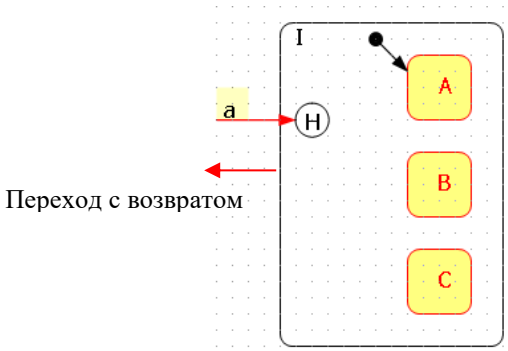

Рисунок 3 Историческое состояние

Заметим, что в диаграмме переход из гиперсостояния I не помечен никаким событием синхронного перехода – это асинхронный переход по прерыванию из любого текущего состояния гиперсостояния I. При этом ожидается асинхронный возврат диаграммы в гиперсостояние I при наступлении в диаграмме некого события a, которое возвращает диаграмму в гиперсостояние I, переход осуществляется во внутреннее состояние H, после чего система безусловно вернётся в то состояние из множества внутренних состояний {A, B, C} гиперсостояния I, которое она покинула в результате асинхронного выхода из гиперсостояния I.

Если же выход из гиперсостояния I был синхронный по событию, то реагирующая система может вновь войти внутрь гиперсостояния I либо в начальную вершину A, ли в другую вершину, явно заданную переходом.

Условные состояния

Диаграмма реагирующей системы допускает, что переход по событию из некоторого состояния не является безусловным. То есть может оказаться так, что в процессе перехода выявляется новое состояние, удовлетворяющее условию, зависящему от результатов действий, выполняемых при выходе из текущего состояния и в процессе перехода. Иными словами, направление перехода можно оценить только уже выйдя из текущего состояния. Для этого в диаграмме используется проходное промежуточное (неустойчивое) *условное состояние*, в котором результаты действия входа из состояния и действия начавшегося условного перехода уже известны, и предикаты переходов для выбора целевого состояния перехода уже можно вычислить "на лету".

На рисунок 4 представлен простой фрагмент диаграммы состояний, описывающей эту ситуацию, когда например, событием является приход сообщения, а реакция на него зависит от содержимого этого сообщения. Если при приходе сообщения (событие a) система находилась в состоянии A, то она перейдёт в состояние C только в том случае, если выполняется условие P. Если условие P не выполнено, система перейдёт в состояние D.

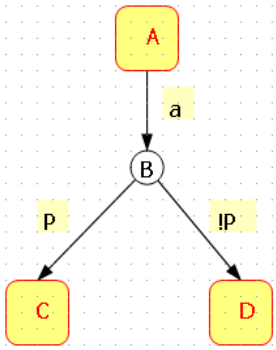

Рисунок 4 Условное состояние

В условном состоянии B система не задерживается, а после анализа сообщения "на лету" выбирает нужное направление перехода в следующее состояние.

Примеры применения диаграмм состояний

Моделирование протокола

В качестве примера рассмотрим применение языка диаграмм состояний и переходов для моделирования протокола доступа и обмена сообщениями рабочих станций высокоскоростной локальной сети (рисунок 5). В соответствии со спецификацией протокола в сети работает множество рабочих станций, и в каждой из них для обмена сообщениями активизирован протокол доступа к сети. Во всех рабочих станциях реализован одинаковый стек сетевых протоколов обмена сообщениями.

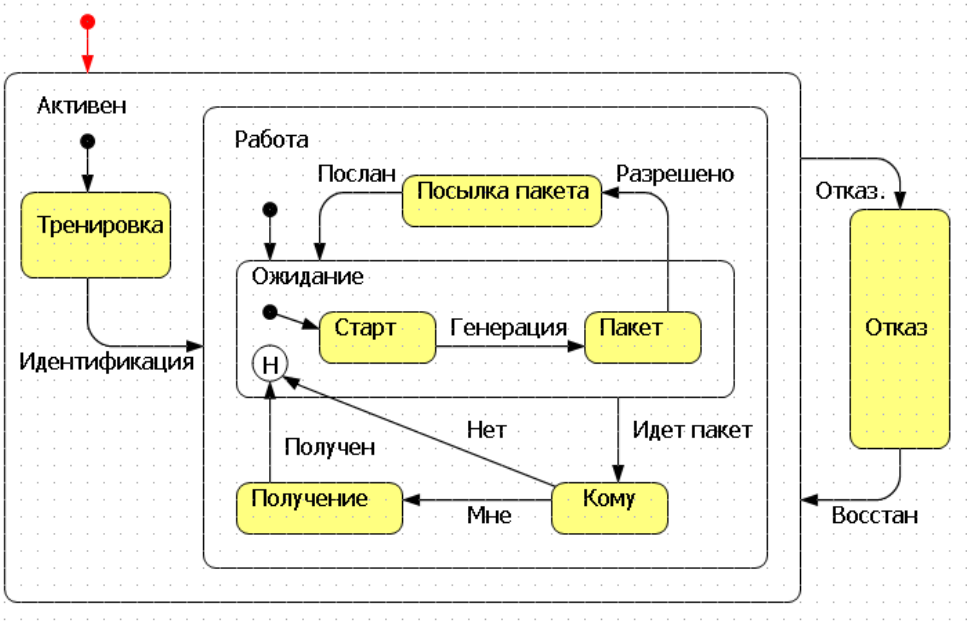

Рисунок 5 Диаграмма состояний процесса доступа к среде протокола IEEE 802.12

Представленная на рисунке диаграмма состояний имеет трёхуровневую иерархическую структуру. На самом верхнем начальном уровне диаграммы (Уровень 1) находятся два состояния – гиперсостояние Активен и элементарное состояние Отказ. На следующем уровне детализации диаграммы, внутри состояния Активен, находятся элементарное состояние Тренировка и гиперсостояние Работа (Уровень 2). На следующем уровне детализации, внутри состояния Работа, находятся три элементарных состояния: Посылка пакета, Кому, Получение, и одно гиперсостояние Ожидание (Уровень 3). На следующем самом нижнем (последнем) уровне детализации диаграммы состояний, внутри состояния Ожидание, находятся

два элементарных состояния Старт и Пакет (Уровень 4).

Протокол доступа каждой локальной станции к сети начинает свою работу, с перехода в состояние Активен на самом верхнем первом уровне диаграммы состояний, за которым сразу следует переход на второй уровень в элементарное состояние Тренировка, так как внутри состояния Активен - на втором уровне диаграммы - ему соответствует начальный переход в элементарное состояние Тренировка. В состоянии Тренировка выполняются операции по идентификации данной рабочей станции в локальной сети, проверке канала, реализующего верхний уровень стека сетевых протоколов. После идентификация станции в сети выполняется безусловный переход Идентификация - диаграмма переходит в состояния Работа, Ожидание и Старт, в котором ожидает поступление пакетов данных, генерируемых пользователем данной рабочей станции для передачи. Когда пакет данных сгенерирован, осуществляется безусловный переход Генерация в состояние Пакет, в котором ждёт от верхнего уровня Работа разрешения на посылку пакета в канал. Когда разрешение получено, выполняется безусловный переход Разрешено в состояние Посылка пакета верхнего уровня. Когда возникает событие завершения посылки пакета, срабатывает переход Послан и диаграмма попадает на текущем уровне в гиперсостояние Ожидание, в результате чего сразу оказывается на нижнем уровне в состоянии Старт.

В любом из двух состояний Старт или Пакет, гиперсостояния Ожидание, процесс синхронной работы диаграммы может быть прерван событием посылки в сеть пакета данных другой рабочей станцией локальной сети. Это событие извещает каждую станцию локальной сети о том, что в сети начал передаваться пакет и потенциальному адресату необходимо подготовиться к его возможному приёму. В результате диаграмма выполняет *асинхронный* переход Идет пакет, который переводит диаграмму в элементарное состояние Кому гиперсостояния Работа из любого элементарного состояния Старт или Пакет гиперсостояния Ожидание. Все локальные станции, перешедшие в состояние Кому, принимают пакет и анализируют сведение об адресате, находящееся в заголовке пакета. После

приёма пакета и анализа заголовка диаграмма либо выполняет переход Нет и возвращается через историческое состояние Н в прерванное состояние, в котором она находилась в гиперсостоянии Ожидание в момент события посылки пакета и срабатывания асинхронного перехода Идёт пакет, либо выполняет переход Мне и оказывается в состоянии Получение, в котором получает пакет данных.

Из любого состояния любого уровня в результате возникновения любого контролируемого события отказа в работе станции диаграмма выполняет синхронный переход Отказ, который переводит диаграмму в состояние Отказ. После анализа отказа и восстановления нормальной работы станции диаграмма состояний выполняет переход Восстан в гиперсостояние Активен и вновь начинает штатную работу с состояния Тренировка.

Этот пример показывает, что диаграммы позволяют наглядно и экономно выразить с помощью графической нотации весьма сложное поведение предмета моделирования.

Моделирования иерархических процессов

Следующий пример иллюстрирует ещё одно применение диаграмм состояний для моделирования сложных иерархических процессов. В качестве предмета моделирования выступает "полёт самолёта" (рисунок 6).

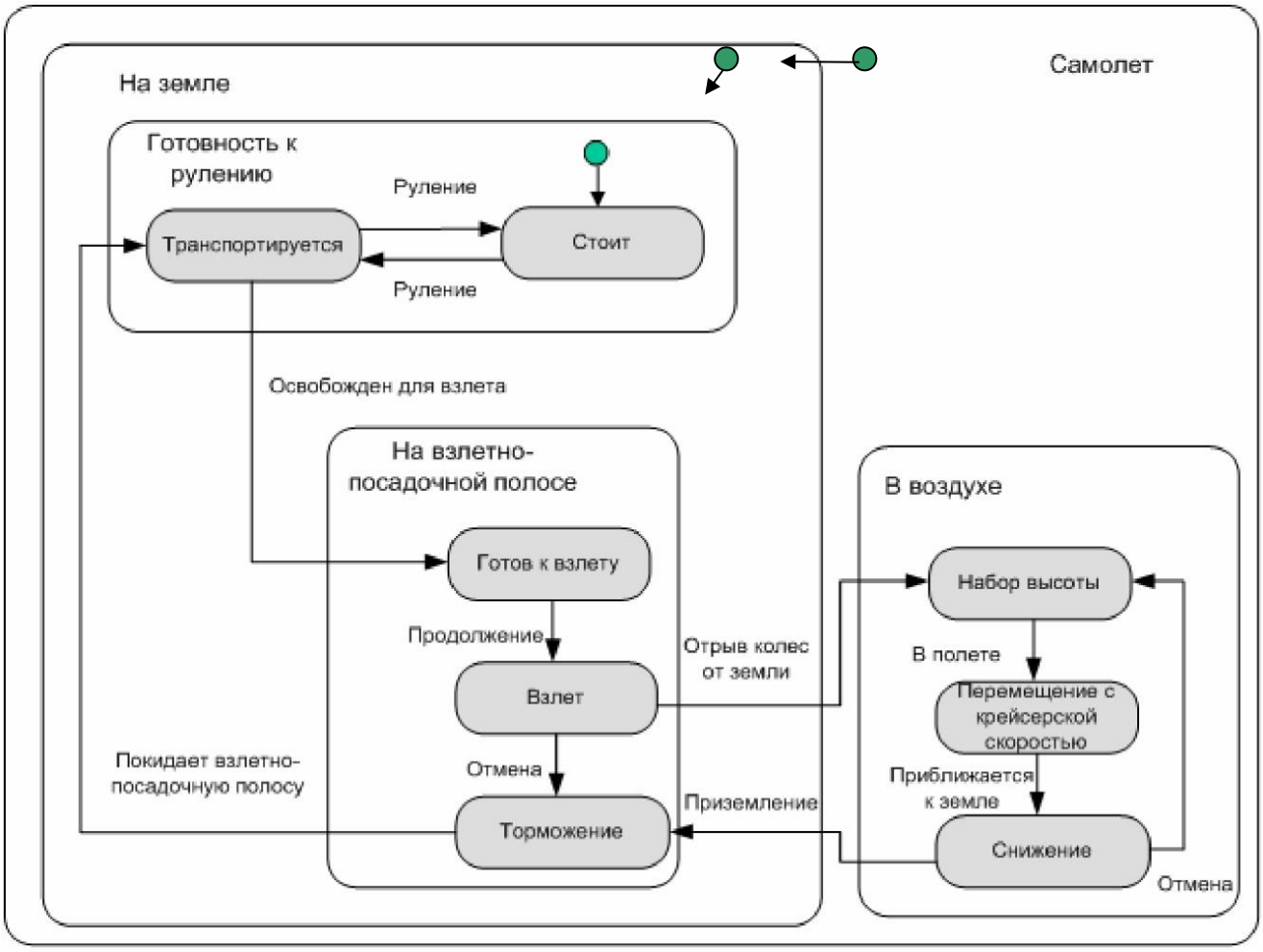

Рисунок 6 Диаграмма состояний полёта самолёта

Корнем иерархии является состояние Самолёт (имя диаграммы), включающее в себя два состояния верхнего уровня - это На земле и В воздухе, между которыми определены переходы Отрыв колёс от земли и Приземление. Внутри состояния В воздухе выделены три независимых состояния: Набор высоты, Перемещение с крейсерской скоростью, Снижение. Внутри состояния На земле выделены состояния Готовность к рулению и На взлётно-посадочной полосе. При этом внутри состояний Готовность к рулению и На взлётно-посадочной полосе также имеются собственные вложенные состояния.

Система переходит в состояние В воздухе, когда колеса самолёта отрываются от земли, а когда колеса касаются земли, система возвращается в состояние На земле. Далее эти состояния детализируются в иерархическом порядке.