Адаптивные расписания как семантическая модель управляющих алгоритмов реального времени

А. А. Тюгашев
Самарский государственный университет путей сообщения
tau797@mail.ru

***Аннотация*. Статья посвящена проблеме моделирования семантики управляющих алгоритмов реального времени (УА РВ). В качестве семантической модели предлагается использовать адаптивные расписания. Подобный подход предоставляет ряд преимуществ перед известными семантическими моделями алгоритмов управления, основанными на состояниях, такими как модель Крипке, сети Петри. В частности, предложенная модель позволяет избежать проблемы 'комбинаторного взрыва' числа состояний. Адаптивность расписания позволяет в достаточной мере отразить семантику управляющего алгоритма за счет моделирования различных возможных 'траекторий' исполнения, зависящих от значений проверяемых при исполнении переменных. В статье приводится пример применения предложенной модели и проблемно-ориентированного языка, построенного на ее основе, в рамках автоматизированных технологий поддержки проектирования и верификации бортового управляющего обеспечения космических аппаратов.**

Ключевые слова: адаптивное расписание; семантика программы; управляющий алгоритм реального времени; формальная верификация; логика управляющих алгоритмов реального времени; CASE инструментарий

I. Введение

В настоящее время широко используются сложные технические комплексы, интегрирующие разнообразное бортовое оборудование и аппаратуру – датчики, агрегаты, исполнительные механизмы, и пр. Примерами могут служить космические аппараты и ракеты-носители, атомные электростанции, автомобили, и т.д. Фактически, подобный сложный технический комплекс представляет собой «систему систем». При этом обычно от данного сложного технического комплекса требуется реализация нетривиального поведения, и/или выполнения некоторого заданного плана. Для достижения данной цели необходимо обеспечить согласованное функционирование всех бортовых устройств, причем человек может быть как задействован в контуре управления, так и нет. Под «согласованным функционированием» подразумевается как согласование выполняемых операций с точки зрения логики управления и физики протекающих процессов, так и во времени (правильная синхронизация). Поскольку необходимо выполнение управляющего алгоритма в режиме реального времени, скорость работы системы управления должна соответствовать скорости протекающих в объекте управления и внешней среде процессов [1–3].

Одним из наиболее ярких и представительных примеров подобного технического комплекса является космический аппарат (КА) [2–5]. В его состав, как правило, входит множество бортовых систем, таких как система электроснабжения, система управления движением, бортовой комплекс управления, телеметрическая система, и т. д. Каждая из систем, в свою очередь, состоит их множества различных по назначению и типу устройств. При этом сегодня даже наноспутники массой до 10 кг оборудованы бортовым компьютером или даже бортовой вычислительной сетью [6]. Сложность ее соответствует сложности КА [6, 7]. Непосредственная реализация логики управления в различных ситуациях осуществляется бортовым программным обеспечением (БПО). Можно констатировать, что, в этой ситуации успех космического полета прямо зависит от качества и надежности БПО, которое, таким образом, является ярким примером «критически важного» программного обеспечения. При этом при создании современной системы управления КА уровни затрат на аппаратную и программную часть соотносятся как 1:10 [3, 6]. Во многих случаях разработка программного обеспечения становится «критическим путем» на сетевом графике работ по созданию ракетно-космического комплекса в целом [3]. При этом все модули БПО должны проходить создании и каждом последующем изменении (а они неизбежны в условиях длительной эксплуатации и совершенствования БПО) ряд обязательных стадий – анализ, проектирование, верификация, испытания, и пр. В результате стоимость БПО постоянно растет. При этом важно отметить, что тестирование в принципе не может дать полной гарантии безошибочности программного обеспечения. В создании БПО участвует множество людей, и существует эффект «испорченного телефона», когда специалисты, разбирающиеся в особенностях управления той или иной бортовой системой, не могут донести их корректно до программистов, воплощающих логику управления в терминах языка программирования – C, C++, Java, и т.п. Еще одной важной причиной сбоев является несовершенство спецификаций, которые применяются в

качестве исходных данных для разработки БПО. Спецификации сами по себе могут быть неполными или содержать противоречия. Увы, подобные ситуации неоднократно приводили к авариям и катастрофам [1, 6, 7]. Когда мы ведем речь о «правильности» программы, весьма важным аспектом является ее семантика. К сожалении, основные известные результаты в области семантического моделирования программ направлены в основном на вычисления, а не управление. И применение известных подходов приводит к тому, что, например, оценка параметров синхронизации становится чересчур сложной или даже невозможной.

Весьма важные особенности отличают управляющий алгоритм от вычислительного. Вычислительный алгоритм ассоциируют с той или иной функцией (в математическом смысле). Корректность подобного алгоритма полностью определяется состоянием программной памяти ЭВМ в момент завершения выполнения программы. В то же время, многим управляющим алгоритмам не удастся сопоставить никакую вычислимую функцию. Их корректность нельзя определять только по состоянию ячеек памяти в тот или иной момент времени, весьма важной является история их поведения на протяжении времени исполнения. Условия, проверяемые в ходе исполнения вычислительного алгоритма, полностью вытекают их входных данных. Значения, проверяемые управляющим алгоритмом, зачастую являются считанными с датчиков параметрами, «внешними» отношению к программе. Кроме того, действия, выполняемые управляющим алгоритмом – это не только преобразования информации, но и выдача управляющих воздействий, которые влияют на протекающие вовне системы управления процессы. Если нам, например, необходимо обеспечить посадку КА на Марс, нам нужно воспроизвести целый комплекс (циклограмму) операций, выполняемых различными бортовыми системами и связанных между собой как с точки зрения логики, так и во времени.

Можно отметить, что исторически основные усилия исследователей со времен машин Тьюринга и Поста, аппарата рекурсивных функций, нормальных алгорифмов Маркова, направлялись на моделирование вычислительных алгоритмов. Подобные алгоритмы легко моделируются с помощью «традиционных» блок-схем, имеющих начало и конец, и состоящих из связанных передачей управления действий, являющихся преобразователями данных (арифметическими и логическими операциями). При этом временные ограничения и таймеры в данных моделях отсутствуют.

В ряде случаев управляющий алгоритм может рассматриваться, как реагирующая система, которая должна некоторым образом обрабатывать входящий поток событий. Основными элементами такого алгоритма являются уже действия. Известны такие семантические модели для реагирующих систем, как (1) модель Крипке; (2) конечно-автоматные модели; (3) сети Петри. Все они основываются на концепции состояния, и страдают от проблемы «комбинаторного взрыва» числа подлежащих анализу состояний при попытке применения их в реальном мире к системам промышленного уровня сложности. Можно отметить следующие основные особенности реагирующих (пассивных по натуре) алгоритмов: (1) типично отсутствие начала и конца, выполняется бесконечный цикл обработки событий; (2) асинхронная натура – отсутствие временных ограничений и таймеров; (3) корректность алгоритма определяется правильностью реакции на происходящие события.

Однако, предметом настоящей статьи являются активные по своей сути, так называемые «управляемые временем» управляющие алгоритмы (УВ УА РВ). Данный вид алгоритмов существенно отличается от реагирующих систем. Управляемый временем алгоритм должен реализовать выполнение некоторого «графика» или «расписания», с действиями, привязанными к заданным моментам времени. В аэрокосмической области широко применяется слово «циклограмма». Важнейшими особенностями УВ УА РВ являются: (1) основные составляющие – действия; (2) в отличие от реагирующих систем, обычно наличествуют начало и конец; (3) имеются жесткие временные ограничения, нарушение коих означает некорректность алгоритма; (4) присутствуют таймеры для количественной оценки и отсчета заданных временных интервалов, иными словами алгоритм имеет синхронную природу; (5) алгоритм корректен, если он выполняет необходимые операции в заданные моменты времени при правильном учете текущей ситуации, отражаемой проверяемыми параметрами (условиями). Пятое условие фактически значит, что семантика алгоритма должна отражать возможность различных вариантов, или «траекторий» его исполнения. Например, УА РВ космического аппарата должен обеспечивать выполнение поставленных задач как при штатной функционировании бортовой аппаратуры, так и в случае отказа того или иного прибора, например, обеспечив переход на резервный.

В тексте программы варианты воплощаются с помощью операторов ‘if’ и ‘case’.

II. Использование адаптивного расписания при семантическом моделировании УА РВ

Мы декларируем, что расписание (циклограмма) может быть использована для представления семантики управляющего алгоритма реального времени, т.е. того, что алгоритмы должен выполнить. В самом деле, при семантическом моделировании УА РВ мы должны учитывать, что многие действия выполняются не мгновенно, а являются протяженными во времени. Например, может потребоваться время для прогрева и перехода некоторого прибора в рабочий режим. Другой пример – для набора КА нужной скорости необходимо обеспечить заданное время работы ракетного двигателя. Соответственно, мы можем указать t_i – время начала действия f_i, и его продолжительность τ_i (рис. 1).

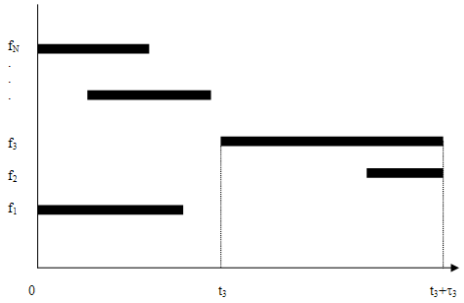

Рис. 1. Расписание в качестве семантической модели УА РВ

Соответственно, расписание

$$S = \{ < f_i, t_i, \tau_i > \}, i = \overline{1,N}$$

может рассматриваться в качестве семантической модели данного класса алгоритмов. Как было подчеркнуто выше, УА РВ должен содержать различные ветви для разных возможных ситуаций (например, штатное функционирование объекта управления, аппаратный отказ, программный сбой, и пр.) [8, 9]. Это приводит к расширению модели – нам необходимо не просто фиксированное расписание, а «адаптивный» вариант, иными словами, мы должны рассматривать набор диаграмм для всех учитываемых ситуаций. При этом ситуация может быть отображена с помощью значений проверяемых в алгоритме переменных. Таким образом, семантика управляемого временем УА РВ может быть представлена, как набор кортежей:

$$УА\ РВ = \{ < f_i, t_i, \tau_i, \overline{l_i} > \}, i = \overline{1,N},$$

где f_i – идентификатор выполняемого действия; t_i – момент начала выполнения действия (целое число); τ_i – продолжительность действия (неотрицательное целое число); $\overline{l_i}$ – логический вектор, содержащий переменные, проверяемые в узлах ветвления алгоритма; заданный логический вектор разрешает выполнение данного действия на указанном временном интервале. Например, логический вектор (α_1=Л, α_2=Н, α_3=И, α_4=Л, α_5=Н) значит, что (1) всего в алгоритме проверяется пять условий (2) данный вектор обуславливает исполнение некоторого действия ложностью (‘Л’) условий α_1 и α_4, истинностью α_3 (‘И’), причем исполнение действия не зависит от значений α_2 и α_5 (‘Н’ значит ‘нейтральный’). Логический вектор является неким аналогом конъюнкта, входящего в ДНФ булевой функции ($\neg\alpha_1.\alpha_3.\neg\alpha_4$), но, в отличие от него, логический вектор включает все условия, проверяемые в управляющем алгоритме. Это важно с точки зрения реализации средств автоматизации анализа [9]. Компоненты логического вектора могут рассматриваться, как аналоги «охраняющих» условий.

Пример. Возможная семантика управляющего алгоритма
{<f_1,0,20,(α_1=T)>,<f_2,0,100,(α_1=F)>,<f_3,20,200,(α_1=T)>,
<f_4,180,50,(α_1=N)>,<f_5,220,15,(α_1=N)>}.

Также мы можем рассматривать набор изображений для каждой ситуации. Сравнивая предложенный подход с моделями семантики, основанными на состояниях, можно отметить следующее. Для приведенного примера можно построить временной автомат с соответствующими переходами между состояниями, однако это построение не является простым процессом. В противоположность этому, предложенная модель не требует никаких дополнений или уточнений. Мы можем сказать, что модель, основанная на состояниях, отражает алгоритм на «микроуровне», а наша модель – на «макроуровне». Переходы между состояниями в явном виде отсутствуют. Получается, одна и та же семантика может быть реализована различными управляющими программами, имеющими разные управляющие графы (в терминологии [4, 9] «логико-временные схемы»). В результате, появляется пространство для проведения оптимизирующих преобразований программ, например, по критерию числа ветвей. Эта ситуация подобна случаю вычислительных алгоритмов, когда одинаковое семантически преобразование данных может быть реализовано разными программами с весьма различной степенью эффективности. Важно отметить, что предлагаемая модель, как это видно на представленных рисунках, весьма легко и наглядно визуализируется, что представляется довольно важным преимуществом с точки зрения восприятия и анализа человеком. Еще одним важным преимуществом является относительная легкость обратного проектирования, или реверс-инжиниринга, когда семантическая модель может быть «извлечена» из существующей программы или ее исходного текста. На основе построенной семантики становится возможным проверить выполнение требований синхронизации действий. Подобный контроль может проводиться с участием человека, на базе специального средства автоматизации.

III. Пример: спецификация и верификация управляющего алгоритма

Представленная модель успешно была использована на практике, например в CASE-системе автоматизации проектирования и верификации бортового программного обеспечения беспилотных космических аппаратов

Рис. 2. Вид экрана средства визуализации спецификации системы ГРАФКОНТ/ГЕОЗ

ГРАФКОНТ/ГЕОЗ [4, 9, 10]. Данная система позволяет поддерживает модельно-ориентированный процесс разработки управляющей программы от стадии спецификации до стадий генерации программы и отладки. В системе применяется специальный проблемно-ориентированный язык, способный описывать ключевые требования с точки зрения синхронизации и логической связи выполняемых УА РВ действий [4, 9]. Язык включает набор формул для отражения требуемых временных связок. Так, $f_1 \rightarrow f_2$ значит, что действие f_2 начинается немедленно по окончании f_1, f_1 CH f_2 значит одновременный старт обоих действий, $(\sim\alpha_3) \Rightarrow f_7$ означает, что выполнение действия f_7 обуславливается ложностью условия α_3. Таким образом, мы, например, можем иметь дело со спецификацией вида: f_1 *CH* f_2 ; $f_1 \rightarrow f_3$; f_4 *CK* f_5 ;$f_3 \rightarrow f_4$;$f_2 \rightarrow f_5$. Система ГРАФКОНТ/ГЕОЗ имеет средства для визуализации подобным образом заданных спецификаций управляющего алгоритма, копия экрана показана на рис. 2. Еще одной важной особенностью представленной модели является легкость ее реализации средствами языков программирования, включающими поддержку списков в качестве базовых структур данных – Лисп, Пролог, и т.д. Действительно, семантика УА РВ представлена как набор (список) кортежей (а кортеж может быть рассмотрен, как неоднородный список), логический вектор может быть представлен в виде списка, и т.п. Одним из главных условий корректного управления является правильная синхронизация выполняемых действий. Например, нас могут интересовать ответы на вопросы: «Выполняется ли системой управления условие, что действие f_1 заканчивается до начала исполнения f_2 ?», «стартуют ли f_1 и f_5 в один и тот же момент времени?», «одними ли и теми же условиями обусловлено исполнение действий f_2 и f_7?», «заканчивается ли выполнение действия f_3 к моменту времени t=500?», «как много параллельных действий исполняется на временной метке t=120, в случае истинности заданного набора условий?». Одним из возможных путей поиска ответов на подобные вопросы является визуализация семантики, построенной по исходным текстам программ, позволяющая производить «быструю оценку» выполнения требований синхронизации буквально «на глаз». Помимо этого, для более точного и глубокого анализа могут использоваться специальные средства автоматизации. Специальный алгоритм обратного проектирования основана на выполнении следующих основных шагов. Во-первых, мы производит реконструкцию логико-временной схемы программы. Это возможно в силу того, что каждая стандартная операция, исполняемая алгоритмом – ветвление, вызов подпрограммы, временная задержка, и пр., сводится к исполнению некоторого набора инструкций. Затем осуществляется обход логико-временной схемы (подходят стандартные алгоритмы обхода дерева). После этого мы будем иметь времена начала и длительности каждого из выполняемых УА РВ действий. Также алгоритм «накапливает» текущий логический вектор, запоминая ветви проходимых условий в конструкциях ‘if’ и ‘case’. На этой основе становится возможным построение (циклограммы) и проведение дальнейшего анализа. Все это поддержано специальными программными средствами, интегрированными с встроенной машиной вывода языка логического программирования Пролог [4, 9].

IV. Заключение

Представлена семантическая модель «управляемых временем» управляющих алгоритмов реального времени. Использование данной модели позволяет проводить автоматизированную оценку исполнения требований синхронизации выполнения действий при управлении сложными техническими комплексами в некоторой степени аналогично известному подходу model checking, однако особенности предложенной модели позволяет избежать проблемы комбинаторного взрыва числа состояний. Приведен пример применения представленной модели в специальной CASE системе поддержки проектирования и верификации бортовых управляющих алгоритмов реального времени для космических аппаратов.

Список литературы

[1] Ахметов Р.Н., Макаров В.П., Соллогуб А.В. Принципы управления космическими аппаратами мониторинга Земли в аномальных ситуациях // Информационно-управляющие системы. 2012. Т. 1, вып 1(56). С. 16-22.

[2] Тюгашев А.А., Железнов Д.В., Никищенков С.А. Технология и инструментальный программный комплекс проектирования и верификации алгоритмов управления реального времени // Элетротехника. 2017. Т. 88, вып. 3, С. 59-64.

[3] Тюгашев А.А., Ермаков И.Е., Ильин И.И. Пути повышения надежности и качества программного обеспечения в космической отрасли // Управление большими системами. Сборник трудов. 2012. Инс-т проблем управления им. В.А.Трапезникова РАН, вып. 39 С. 288-299.

[4] Тюгашев А.А. Интегрированная среда для проектирования управляющих алгоритмов реального времени // Известия Российской академии наук. Теория и системы управления. 2006. № 2. С. 128-141.

[5] J. Eickhoff, Onboard computers, Onboard Software and Satellite Operations. An Introduction. Springer-Verlag Berlin Heidelberg, 2012.

[6] Козлов Д.И., Аншаков Г.П., Мостовой Я.А., Соллогуб А.В. Управление космическими аппаратами зондирования Земли: Компьютерные технологии. М.:Машиностроение, 1998.

[7] Хартов В. В. Автономное управление космическими аппаратами связи, ретрансляции и навигации // Авиакосмическое приборостроение. 2006. № 6. С. 29-33.

[8] Салмин В.В., Филатов А.В., Ткаченко И.С., Тюгашев А.А., Сопченко Е.В. Вычислительный алгоритм формирования программного движения в программном повороте малого космического аппарата // Вестник Самарского государственного аэрокосмического университета им. академика С.П. Королёва. Т. 14, № 2 (2015)/ С.9-19.

[9] А.А. Калентьев, А.А. Тюгашев. ИПИ/CALS технологии в жизненном цикле комплексных программ управления. Самара: Изд-во Самарского научного центра РАН, 2006. 254 с.

[10] Тюгашев А.А. Автоматизация спецификации, верификации и синтеза управляющих программ реального времени с применением логического и алгебраического подходов // Мехатроника, автоматизация, управление. 2007. № 7. С. 46-51.

[11] A.A. Kalentyev, A. A. Tyugashev, A.A. Bogatov, A.A. Shulyndin, “Visual toolset for real-time onboard programs verification support”, in Proc. Program Semantics, Specification and Verification: Theory and Applications (PSSV 2011), St.Petersburg, Russia, June 12th 2011, pp. 1.